For example, given that the number of rows and the number of columns are to be M and N respectively, an encoder specifies values of M-1 for the rc and N-1 for the cc. The decoder, therefore, adds one to both the rc and cc resulting in the creation of a window containing M rows and N columns.

Window height may be specified to have up to 15 rows (rc 0 through 14). Decoders should be capable of interpreting and displaying windows with rc in the range 0 - 14, though some legacy decoders may not be able to display more than 12 rows.

Windows shall have between 1 and 32 columns of characters (cc values 0 through 31).

8.4.7 Window Row and Column Locking

Prior to CEA-708-C, parameters row lock (rl) and column lock (cl) in the DefineWindow (DFx) command were used for control of window row and column locking, providing alternatives in growing and shrinking caption windows. CEA-708-C specifies a single method for window growing and shrinking, and in section 8.10.5.2 these parameters are reserved. Future versions of CEA-708-C may allow use of these parameters, but decoders compliant with CEA-708-C shall ignore the values, and assume rows and columns are locked. Future versions of CEA-708-C may allow the optional decoder implementation of windows with unlocked rows and columns, and may use these syntax elements.

Change of font style changes the dimensions of characters displayed in a window. The same might be true of changes in other pen attributes, such as italics, offset and underline. Windows shall be large enough to fit the text they contain, rendered in the transmitted font as modified by viewer selections. Windows should be initially sized to fit anticipated characters in the effective font size. If a change in pen attributes changes the window size, a decoder may be required to modify a displayed window in such a way that it grows beyond the safe title area. The following sections provide guidance in this area, and in the area of windows that overlap due to pen selections.

8.4.7.1 Effects When Choosing Smaller Character Size

Figure 16 illustrates effects on a window containing STANDARD-sized caption text when a viewer selects the SMALL font.

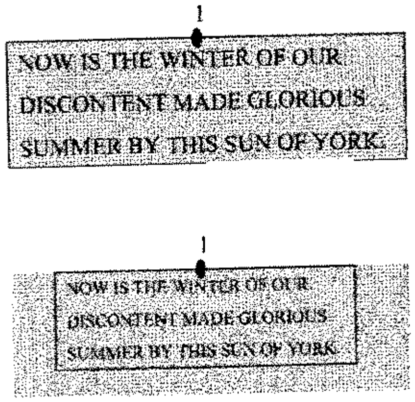

Figure 16 Examples of Caption Window Shrinking when User Selects Small Character Size

In Figure 16, the shaded rectangle indicates the size of the original window. The lower display shows how the window should be modified when the viewer selects SMALL Character Size. The window in this example was specified with *anchor point = 1*. The caption text and window shrink to a size in direct proportion to the original window. The caption contains the same number of lines and columns as the original, and the same words appear on the same line.

If the viewer has selected SMALL character size, other caption windows and video beneath the original window may be revealed.

8.4.7.2 Effects When Choosing Larger Font

Figure 17 illustrates the effects on a window containing STANDARD-sized caption text when a viewer selects the LARGE character size.

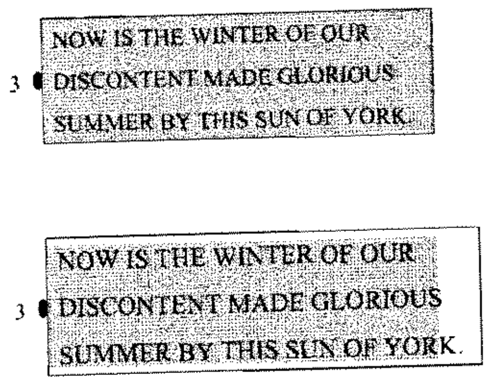

Figure 17 Examples of Caption Window Growing when Going to Larger Font

In Figure 17, the shaded rectangle indicates the size of the original window. The window in this example was specified with *anchor point* = 3. The caption text and window grow to a size in direct proportion to the original window. The caption contains the same number of lines and columns as the original, and the same words appear on the same line.

Windows should grow based on the recommendations above. If a change cannot be made without exceeding the safe title area, the decoder may modify any pen attribute of any characters in the window to make the window fit.

Visibility of the caption message is the primary goal of closed captioning. Fonts, emphasis and special effects are secondary to information display. For this reason, making the text visible should take priority.

This recommendation could cause LARGE characters to revert to STANDARD, italic to non-italic, or other pen changes. It should only be used as a last resort. It is known that use of STANDARD, monospaced font for all characters produces a window that fits within the safe title area. If no other methods of providing a visible caption provide an understandable caption display, this recommendation can provide a solution.

> NOTE—There are some situations with no solution, such as use of 32 LARGE characters on a 4:3 display. While the modifications described above are preferred, decoders are not required to modify pen attributes or window positions to accommodate characters in this situation, and could disregard that window.

8.4.8 Word Wrapping

Prior versions of CEA-708-C used parameter *wordwrap* in the SetWindowAttributes command for control of word wrap. This option is no longer available. In Section 8.10.5.8, the parameter *wordwrap* is specified to always be 0. Automatic cursor movement between rows was previously possible for windows with the wordwrap (ww) parameter set to 1. Decoders compliant with CEA-708-C shall ignore this parameter in the DefineWindow and SetWindowAttributes command, and not wrap or hyphenate words at the end of a row. The change of cursor position between lines within a window may only be performed through use of CR or FF control codes, or the SPL Command Code. Characters entered into a line when the cursor is at the final character position shall either replace the final character or be discarded. Caption providers shall set the "wordwrap" parameter to '0' in all windows. The paramter wordwrap = 0 may change in future versions of CEA-708-C.

8.4.9 Window Text Painting

This section defines window text painting parameters and the interactions between them.

8.4.9.1 Justification

Caption text shall be formatted within the window based upon the justification type. The nomenclature for the justification types is based on standard English left-to-right Print Direction (see Section 8.4.9.2), and should be interpreted as follows:

a) **Left Justification**—Text is justified to the start of the row, e.g., the left side for left-to-right print direction, the bottom for bottom-to-top print direction, and so on.

b) **Right Justification**—The opposite of left justification. The text is aligned to the end of the row.

c) **Centered Justification**—The text is centered horizontally when the print direction is left-to-right or right-to-left, and centered vertically when the print direction is top-to-bottom or bottom-to-top.

d) **Full Justification**—The text is aligned to the left and right margins when the print direction is left-to-right or right-to-left, and to the top and bottom margins when the print direction is top-to-bottom or bottom-to-top.

8.4.9.1.1 Effect of Justification Values and Clearing of Windows & How Justification Affects Display Timing

The justification of a window, as set by the Set Window Attributes (SWA) command, can effect the display time of caption text sent to that window.

8.4.9.1.2 Display for Various Justification Settings

When the window attribute Justify is set to LEFT, characters written to a non-hidden, left-justified window shall be displayed immediately upon receipt by the decoder. Multi-byte characters shall be displayed when the last byte of the character is received.

When the window attribute Justify is set to RIGHT, CENTER, FULL, characters written to a non-hidden window with Right, Center, or Full justification may be displayed immediately upon receipt by the decoder, or they may be cached until an entire row has been received and then displayed. A "row completion indicator" is indicated as described in Section 9.10.1. When sending data to visible windows with centered or right justified text, caption authors should signify "row completion" as soon as possible to provide consistent caption viewing on cached and non-cached decoders.

The values above apply to "Left-to-right" text windows.

8.4.9.1.3 Change of Window Attributes in a Displayed Window

A new SWA command may be received for a currently displayed window.

It is recommended that any text received prior to the new SWA command not be redisplayed, and only text received subsequent to the new SWA command be written with the new alignment. Caption authors should not rely on this operation, but should issue a ClearWindows, Delete Windows or Reset command to empty the window being redefined.

8.4.9.2 Print Direction

The Print Direction parameter specifies in which direction characters will be written on a (horizontal or vertical) caption row (left-to-right, right-to-left, top-to-bottom, or bottom-to-top).

Table 21 defines how the cursor should move after drawing a character (when Justification is "Left" – see Section 8.4.9.1) or after receiving a carriage return for each combination of Print Direction and Scroll Direction (see Section 8.4.9.3). Combinations of Print Direction and Scroll Direction that are not listed in this table are not permitted.

Print Direction	Scroll Direction	Cursor Movement	Carriage Return Behavior
Left -> Right	Top-> Bottom	increment column	decrement row, column=0
Left -> Right	Bottom->Top	increment column	increment row, column=0
Right->Left	Top-> Bottom	decrement column	decrement row, column=max
Right->Left	Bottom->Top	decrement column	increment row, column=max
Top-> Bottom	Left -> Right	increment row	decrement column, row=0
Top-> Bottom	Right->Left	increment row	increment column, row=0
Bottom->Top	Left -> Right	decrement row	decrement column, row=max
Bottom->Top	Right->Left	decrement row	increment column, row=max

Table 21 Cursor Movement After Drawing Characters

Figure 18 shows how a 3-row caption would appear using various justifications (see Section 8.4.9.1), Print Directions and Scroll Directions (see Section 8.4.9.3).

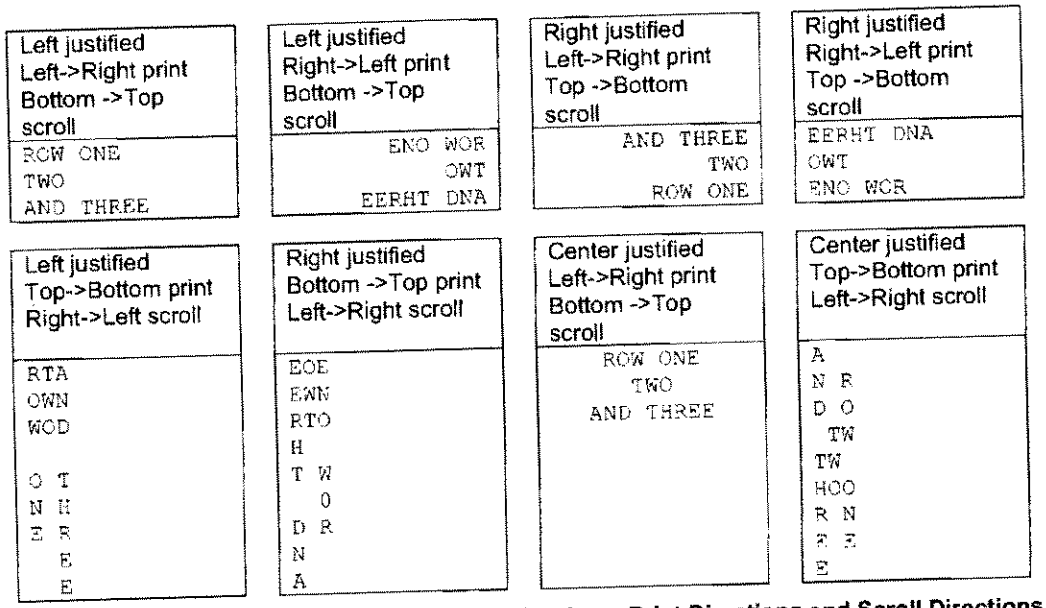

Figure 18 Examples of Various Justifications, Print Directions and Scroll Directions

8.4.9.2.1 Effect of Changes in Print Direction by SetWindowsAttributes & SWA Interaction

A new SWA (Set Window Attributes) command may be received for a currently displayed window. The new SWA command may change the Print Direction attribute for the window. CEA-708-C does not define how a decoder should treat existing text in the window upon a change of print direction.

8.4.9.2.2 Print Direction Changes

A caption author is allowed to change the Print Direction attribute for a window to which text has already been written. It is recommended that upon receiving an SWA command with a new value for Print Direction compared with the direction of the last character sent to the window, all existing text in the window be cleared, and the cursor placed at the default cursor position (as if the window had just been defined).

Any text received prior to the new SWA command should be removed from memory and not be redisplayed. Text received subsequent to the new SWA command should be written in the new print direction.

Caption authors should not rely on this operation, but should issue a ClearWindows command to empty the window being redefined.

8.4.9.3 Scroll Direction

The Scroll Direction parameter specifies in which direction the text will scroll (left-to-right, right-to-left, top-to-bottom, or bottom-to-top) when carriage returns are encountered. For example, NTSC style roll-up captions are achieved by specifying left-to-right printing with bottom-to-top scrolling.

Horizontal scrolling is allowed only with vertical print directions, and vertical scrolling with horizontal print directions (see Section 8.4.9.2). For example, left-to-right scrolling may be used with top-to-bottom or bottom-to-top printing, but not with left-to-right or right-to-left printing.

8.4.9.3.1 Effect of Scroll Direction Changes in SetWindowAttributes

A new SWA (Set Window Attributes) command may be received for a currently displayed window. The new SWA command may change the Scroll Direction attribute for the window. The standard does not define how a decoder should treat existing text in the window upon a change of scroll direction.

8.4.9.3.2 Scroll Direction Changes

A caption author is allowed to change the Scroll Direction attribute for a window to which text has already been written. It is recommended that upon receiving an SWA command with a new value for Scroll Direction, all existing text in the window be cleared, and the cursor placed at the default cursor position (as if the window had just been defined). Any text received prior to the new SWA command should be removed from memory and not be redisplayed. Text received subsequent to the new SWA command should be written using the new scroll direction.

Caption authors should not rely on this operation, but should issue a ClearWindows command to empty the window being redefined.

8.4.9.4 Text Painting Attributes

A 2-line ticker-tape effect can be accomplished with the following parameter settings: print direction set to "top-to-bottom" with scroll direction set to "right-to-left", and window size set to 2 rows and X columns. The initial pen location is set to the upper right corner of the window. One character from row 1 is sent, followed by 1 character from row 2, and a carriage return, and so on. The first character for row 1 will appear on the screen by itself. With top-to-bottom print direction, the 1st character for row 2 will appear beneath the first row 1 character. The carriage return is sent, creating a new column and causing the existing 2 characters to scroll to the left and the next row 1 character to appear to the right of the existing row 1 character.

8.4.10 Window Display

Caption text can be written to a window whether it is currently displayed (i.e., visible) or not. Writing a whole caption to a non-displayed window and then sending a **DisplayWindows** command will cause the caption to "pop-on" (if the SNAP display effect has been set for the window). To achieve successive "pop-on" type captions, two windows are required (just as displayed and non-displayed memory buffers are used in NTSC captioning), with the required sequencing of **DisplayWindows** and **HideWindows** commands to maintain the effect.

When a window is displayed or hidden, a Display Effect parameter can be specified which controls whether the window snaps on/off ("pop-on"), fades on/off, or wipes on/off. The rate of fade on/off may be a manufacturer's option, but an effect speed can be specified with the **SetWindowAttributes** command. The direction (left-to-right wipe, right-to-left wipe, top-to-bottom wipe, and bottom-to-top wipe) in which a window wipes on/off can also be specified with this command.

8.4.11 Window Colors and Borders

The area within a window can have different characteristics regarding color and opacity. Windows can be clear, or filled with a specified color. A window can be filled with an opaque color from a color palette (see Section 8.8). The colors within a window can have various effects associated with them. The window can be transparent (i.e., clear - that is, letting the underlying video show through), translucent (i.e., the underlying video is passed through a fixed level of color background filtering so that underlying video may partially show through the window), solid (opaque with a nominal level of saturation), or flashing (alternating transparency and opacity at the nominal, or solid, level of saturation).

Windows may have no borders, or be bounded by an enclosing border. Borders may be raised, depressed, uniform, or drop-shadowed (left or right). Manufacturers have a bit of latitude in how these border effects are achieved. A uniform border may appear as a single line surrounding the window. Raised, depressed and shadowed borders can be used to achieve 3-dimensional effects.

8.4.12 Predefined Window and Pen Styles

Colors, text painting, effects, and border types can be customized with the **SetWindowAttributes** and **SetPenAttributes** commands. However, the caption provider may wish to use predefined standard windows styles. A set of predefined styles will be hard stored in receivers. This set will anticipate the most widely used types of caption windows in order to conserve caption channel bandwidth by eliminating the need to transmit superfluous **SetWindowAttributes** and **SetPenAttributes** commands.

Predefined window and pen styles can be specified by the *window style* and *pen style* ID parameters in the **DefineWindow** command. See Section 9.12 for the defined "predefined window and pen styles".

8.5 Caption Pen

The caption text within a window is written with a *pen*. A pen governs the size, font, colors, and styles of the text within a window. Pen attributes are specified separately from window attributes through the use of the **SetPenAttributes** command. A window may contain text with more than one combination of pen attributes (i.e., different fonts, colors, etc.). Pen characteristics remain constant for a window unless specifically changed (via a subsequent **SetPenAttributes** command) by the caption provider. Pen attribute changes only affect text written after the change (i.e., the **SetPenAttributes** command does not change existing text within a window) and until a new **SetPenAttributes** command is encountered.

8.5.1 Pen Size

CEA-708-C specifies three pen sizes, entitled SMALL, STANDARD and LARGE, for font styles supported by a caption decoder. The pen size intended by the caption author is encoded in the captions, but viewers may override the pen size (i.e. the size of the characters) to be viewed on the decoder as desired.

This section defines limits on the height and width of characters of each pen size, as a fraction of the safe title area. The same safe title area dimensions shall be used for defining character dimensions as are used when mapping screen coordinates for window position, as described in Section 8.2.

8.5.1.1 STANDARD Pen Size

The height of the tallest STANDARD character, in any implemented font, shall be no taller than 1/15 the height of the safe title area. The width of the widest STANDARD character, in any implemented font, shall be no wider than 1/42 or 1/32 the width of the safe title area, for 16:9 and 4:3 display formats respectively. The decoder may use proportional font if all characters are displayed. For other display aspect ratios, the number of characters that can fit on one row shall be 32 times the ratio of the screen width to a 4:3 display. The width of each character will be the inverse of this number multiplied by the width of the safe title area in pels.

8.5.1.2 LARGE and SMALL Pen Size

LARGE pen size shall be implemented such that the widest character in any implemented font is no wider than 1/32 of the width of the safe title area for 16:9 displays. Aside from this constraint, LARGE and SMALL pen height and width dimensions are at the discretion of decoder manufacturers. It recommended however, that choices be such that the character heights and widths are integral numbers of lines and pels. This is to allow flexibility to design optimally legible characters, furthermore:

a) Characters in the LARGE pen size should be no wider than 42/32 times the width of characters in the STANDARD pen size, rounded off to the nearest integral pel value. The height of LARGE characters should be no taller than 42/32 times the height of STANDARD characters, rounded off to the nearest integral line value.

b) Characters in the SMALL pen size should be no narrower than 32/42 times the width of characters in the STANDARD pen size, rounded off to the nearest integral pel value. The height of SMALL characters should be no shorter than 32/42 times the height of STANDARD characters, rounded off to the nearest integral line value.

8.5.1.3 Safe Title Area

For defining the window coordinate grid and character cell sizes, the recommended safe title area is 80% of the production aperture size. This does not produce dimensions that are an integer number of pels and lines for most display aperture sizes. In practice, decoders should use a safe title area that is reduced slightly such that the dimensions of STANDARD pen size characters have integer values.

Table 22 shows production aperture values for common display formats, recommended safe title areas, and resultant recommended sizes of the STANDARD, LARGE and SMALL character dimensions for a fixed-width font. The values for LARGE and SMALL assume that the characters are scaled proportionally in both dimensions.

Production Aperture	**Aspect**	**Safe Title Area**	**STANDARD**	**LARGE**	**SMALL**
(W x H)	Ratio	(W x H)	(W x H)	(W x H)	(W x H)
1920 x 1080	16:9	1470 x 825	35 x 55	42 x 66	28 x 44
1280 x 720	16:9	1050 x 600	25 x 40	30 x 48	20 x 32
720 x 486	4:3	480 x 375	15 x 25	18 x 30	12 x 20

Table 22 Safe Title Area and Recommended Character Dimensions

For aperture dimensions not indicated herein, decoder manufacturers are free to use their own judgment in the specification of safe title area, window coordinate grid and character sizes, pursuant to the following:

a) Reasonable safe title area dimensions shall be chosen.
b) The safe title area shall be divided such that 15 rows of text in the STANDARD pen size can be accommodated.
c) The window coordinate grid shall be chosen such that each STANDARD character is five times the width of a window coordinate grid cell and five times the height of a window coordinate grid cell.
d) Dimensions of LARGE and SMALL character sizes shall be chosen in accordance with the rules and recommendations in section 8.5.1.2.

8.5.1.4 Authoring Captions With Multiple Displayed Windows

There is an implied relationship between window coordinate grid cells, as defined in Section 8.2, and STANDARD character dimensions. This relationship is not explicit, and is not mandatory in caption decoders. Caption decoders are free to choose character dimensions for all three pen sizes, subject to the rules of CEA-708-C. Furthermore, decoder manufacturers were previously free to choose arbitrary values for character cell height and width. Consequently, there is not guaranteed to be any relationship between character cells and window coordinate grid cells in caption decoders.

Decoders may also allow the viewer the option of overriding the intended pen size chosen by the caption author. This may result in windows that were intended to be adjacent instead becoming overlapping at display time. Decoders may support overlapping windows, relocate windows to be non-overlapping, or ignore and not display overlapping windows, as described elsewhere in this standard.

These points should be of particular interest to caption authors; it is generally not possible to author captions with multiple adjacent, non-overlapping windows because of the variety of decoder display options.

8.5.2 Pen Spacing

Monospacing and proportional spacing are inherent attributes of the font chosen to write caption text. They cannot otherwise be controlled by either the caption provider or the user.

8.5.3 Font Styles

Caption providers may specify 1 of 8 different font styles to be used to write caption text. The styles specified in the "font style" parameter of the **SetPenAttributes** command are numbered from 0 through 7.

The following is a list of the 8 recommended font styles. For information purposes only, each font style references one or more popular fonts which embody the characteristics of the style:

0 - Default (undefined)
1 - Monospaced with serifs (similar to Courier)
2 - Proportionally spaced with serifs (similar to Times New Roman)
3 - Monospaced without serifs (similar to Helvetica Monospaced)
4 - Proportionally spaced without serifs (similar to Arial and Swiss)
5 - Casual font type (similar to Dom and Impress)
6 - Cursive font type (similar to Coronet and Marigold)
7 - Small capitals (similar to Engravers Gothic)

Implementation of these font styles by decoder manufacturers is optional. Font styles which are not supported in a decoder should be displayed in an available font which is most similar to the requested font style in the **SetPenAttributes** command.

8.5.4 Character Offsetting

Characters can be positioned relative to the row baseline in one of three ways: subscript (offset vertically downward), superscript (offset vertically upward), or normal (no offset).

8.5.5 Pen Styles

Pen styles can be italicized and/or underlined.

8.5.6 Foreground Color and Opacity

The foreground color and opacity of the caption characters can be specified by the caption provider. The character foreground opacity can be set to transparent (i.e., showing underlying video), translucent (i.e., showing a filtered level of underlying video), solid, or flashing.

8.5.7 Background Color and Opacity

Characters are individually contained within a small, rectangular background box. These background boxes have color and opacity attributes which may be specified separately from character foreground attributes.

8.5.7.1 Changing Character Background Color

Although the character background color can be set for each individual character, caption service providers should not normally use this attribute to create words with multiple color backgrounds.

This recommendation is designed to insure legibility of caption information. If colors are changed several times within a word, it can cause unwanted interactions with window background colors and with italic and cursive fonts. If the window background color is varied to create different background colors for each character, word, or word group, it may interfere with kerning, producing overlapping background colors or uneven character spacing.

Figure 19 is an example of italic characters. In the first row, kerning is used to balance character spacing. The second row illustrates individual character background colors. The third row is an example of italic characters without kerning.

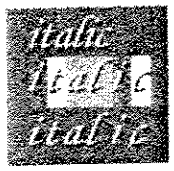

Figure 19 Character Background Color Examples

8.5.8 Character Edges

The color attributes of the edges (or outlines) of the character foregrounds may be specified separately from the character foreground and background. Edge opacities have the same attribute as the character foreground opacities. The type of edge surrounding the body of the character may be NONE, RAISED, DEPRESSED, UNIFORM, or DROP_SHADOWED.

8.5.8.1 Transparent Edges and Character Legibility

CEA-708-C allows considerable latitude in rendering of caption text including decoder implementations that produce confusing and unreadable captions. This section provides guidance on the use and implementation of backgrounds, fills, and transparent spaces.

CEA-708-C specifies the syntax of creating edges, backgrounds, fills and borders around characters, but does not define the meaning of the syntax, leaving this up to the decoder manufacturer's sense of "reasonable". Implementations could cause displayed text to be hard to read, due to interactions of the character pixels and the video background. This was solved in CEA-608-C Section 3.3.2 by requiring the "box" to include a blank character before and after each line of text, isolating the character pixels from the video. This keeps the left and right edges of characters surrounded by background color.

Other techniques can be used with advanced captions. CEA-708-C does not prohibit specifying the "small, rectangular background box" (Section 8.5.7) to extend into the empty, adjacent character cell (which would normally be filled with "background (bg) color"). In many situations, this will improve legibility. CEA-708-C also allows other methods, including font design with a complete frame of bg color surrounding the fg color character pixels. CEA-708-C, therefore, provides more flexibility than CEA-608-C, which requires that the border be exactly one cell's width of background-color box.

Decoder manufacturers should implement non-transparent backgrounds in such a way that the character's legibility is maintained even when the character occurs at the beginning or end of a line. This can be done by extending the background box (described in CEA-708-C Section 8.5.7) beyond the character cell, by inserting a background-filled character space before the first character in a line and after the final character in a line, or by any other means that provides good legibility. This is consistent with recommendations in CEA708-C Sections 9.18 and 9.21. The use of background-filled, as opposed to "fill color" filled improves legibility when "fill opacity" is not SOLID.

Decoder manufacturers should implement TRANSPARENT backgrounds and borders as transparent. This may include the character edges if the "border type" is set to NONE. Character edges may include transition areas surrounding the character pixels, which may be implemented to reduce ringing or other edge effects.

Decoder manufacturers should implement the transparent space as a completely transparent cell, except for transition areas at the beginning and end of a block of transparent cells, which may be implemented to reduce ringing or other edge effects.

Caption service providers should be aware that the use of FLASH, TRANSLUCENT or TRANSPARENT backgrounds may reduce legibility, and should be used with care. This is also true of non-SOLID foregrounds

Caption service providers should be aware that transparent spaces may cause adjacent characters to be less legible, and may wish to improve legibility by inserting edging characters, such as non-transparent spaces into the caption stream before and after a block of transparent spaces.

8.5.8.2 Character Edge Types

Decoder manufacturers may design a text rendering algorithm or circuit to produce font edge types, or store fonts with each of the edge types. The manufacturer may control the appearance of these attributes, but may follow the convention shown below. This definition of character edge attributes follows a convention commonly used by industry. See Figure 20.

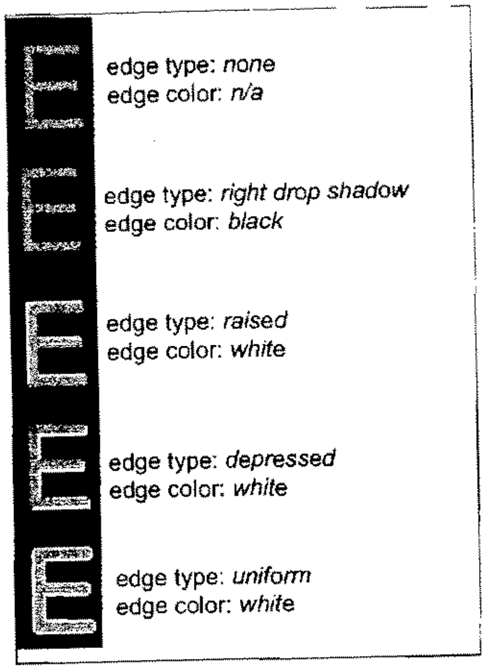

Figure 20 Edge Type Examples

8.5.9 Caption Text Function Tags
Caption text can be tagged with a marker indicating the type of text content that is being encoded. This tagging is performed by caption providers via the **SetPenAttributes** command. With this command, caption text can be tagged with any one of the following qualities:

0 - Dialog (normal words being spoken by characters in the programming)
1 - Source or speaker ID (name of the speaker, or a description of the source of a sound)
2 - Electronically reproduced voice (spoken audio heard by the characters in the drama coming from a phone, radio, PA, etc.)
3 - Dialog in a language other than the drama's primary language
4 - Voiceover (narration or other disembodied voice NOT heard by the characters in the drama)
5 - Audible Translation (voice of a disembodied translator NOT heard by the characters in the drama)
6 - Subtitle Translation (text showing a translation into the primary language of the drama)
7 - Voice quality description (description of a voice quality)
8 - Song Lyrics (words being sung)
9 - Sound effect description (a description of a nonverbal sound or music heard by the characters in the drama)
10 - Musical score description (a description of background music NOT heard by the characters in the drama)
11 - Expletive (an interjectory word or expression, possible profane or harsh)
12 to 14 - (undefined)
15 - Text not to be displayed (reserved for future use by a text-based control and information channel within the caption text stream; e.g., hypertext, related non-caption program information)

In the previous list, all of the tagged text is to be displayed in the current caption window as indicated, except for tag 15 ("Text not to be displayed"), and optionally, tag 11 ("Expletive").

Decoder manufacturers have the flexibility to display tagged text in an number of ways desired in order to enhance the caption viewing experience. For example, "Source or speaker ID" text may always be automatically displayed in italics by the decoder. This may be a feature which the viewer can enable and disable.

If no other information is available, decoders are to default caption text to the "Dialog" tag (0). That is, if caption text is received for a window when no **SetPenAttributes** command has been received, the text is to be treated as "Dialog".

8.5.10 Pen Attributes

Section 8.5.9 allows a text tag to imply particular values for other SPA parameters. The example shows how the "italics" value might be forced on when "Source or speaker ID" is specified in text tag, but underline, edge type, font or even pen size could be attributes of a text tag.

Further, Section 9.15 shows that characteristics such as "italics" may be set by substitution of a font selection.

This enables determination of which attributes the decoder sets, either as a manufacturer preset, or as a user option; however, the SPA command includes values for those attributes that may conflict with the decoder's settings.

The principle applied here is: "The customer is always right". In captioning specification, the farther down the transmission chain a parameter is set, the higher priority it gets.

8.5.10.1 Priority of User Set Values for Pen Attributes

If the user defines the pen size, color, font or any other attribute for caption display, that attribute should be used. It should not be overridden by the SPA command or by a decoder manufacturer defined text tag.

These recommendations give the user the highest priority setting caption appearance. This serves the colorblind, visually impaired, and other communities with special visual requirements. The only exception to this rule occurs when the display itself cannot render the text with the chosen attributes, as in the case of long caption lines in LARGE fonts (see Section 8.4.7.2).

8.5.10.2 Priority of Text Tags over Individual Attributes

If the decoder defines certain attribute values for a text tag, then those values should be used, and conflicting values in the SPA command should be ignored.

These recommendations give the decoder the next highest priority in setting caption appearance. This allows specific choices to be implemented that match the decoder specific display characteristics.

For example, if a decoder's "casual" font is more readable than "cursive", that font could be chosen for "Voiceover". Another decoder manufacturer may have chosen different fonts, and determine that "cursive" is more readable. This manufacturer may choose the "cursive" font for the "Voiceover" text tag.

8.5.10.3 Priority of Font over Font Attributes

If the decoder substitutes a font for a particular font style, and the substitute font requires a particular attribute, then that attribute should be used, and conflicting values in the SPA command should be ignored.

8.5.10.4 Special Properties of Text Tag 0—Dialog

Decoder manufacturers should not define, or allow the user to define parameters specifically for "0 - Dialog" text tag. Text displayed with text tag 0 should use the other attributes defined by the other parameters of the SPA command.

This recommendation allows the caption producers a method to control display attributes individually, using the other SPA parameters. Caption producers are reminded that the display attributes are still subject to viewer overrides.

8.6 Caption Text

Caption text is always written to the current caption window with the attributes set with the preceding **SetPenColor** and **SetPenAttributes** commands. There is no specific Text Write caption command; any displayable characters or codes from the G0, G1, G2, or G3 code sets in Service Blocks which are not part of any caption command are considered text to be written to the current window.

Caption text sequences shall be terminated by either the start of a new DTVCC Command, or with an ASCII **ETX** (End of Text) (0x03) character when no other DTVCC Commands follow. This requirement aids decoders in processing text sequences when text spans multiple service blocks.

8.7 Caption Positioning

The starting row and column position of an individual character or set of characters may be specified at any time via the **SetPenLocation** command. The position of subsequent characters written after a location is specified depends upon the Print Direction and Scroll Direction specified for a window.

Character positions specify memory locations within the internal buffer holding the window text, and thus, do not specify physical locations on the screen. Screen locations are dependent on a multitude of pen and window attributes (e.g., character size, character spacing, justification, and print direction), and whether or not the font is proportionally spaced.

8.7.1 Location within Internal Buffer

The locations specified in SetPenLocation should be those within the "internal buffer." That is, font size, proportional fonts or word wrap does not affect them. This brings order to the several uses of the word column and row. Displayed locations are determined by decoder font, font size, and customer control.

8.7.2 Location (0,0)

SetPenLocation should address a location in the internal buffer, which is "row count"+1 lines long, with "column count"+1 characters per line. The leftmost location in the topmost row is (0,0).

This clarifies the meaning of "row count" and "column count" and defines the location of (0,0).

8.7.3 Caption Line Lengths

Caption providers should not create captions with the number of characters on a line greater than the number of columns in a window as defined in Define Window. Carriage Returns (CR), which should move the cursor to the start of the next row, should be used. Decoders may ignore all characters beyond the last column of a row.

SetPenLocation has no way of identifying locations on an extended line. Therefore, extended lines should not be used.

8.8 Color Representation

Foreground and background colors are specified in the Caption Commands as combinations of the red-green-blue color triad. Two bits are specified for each red, green, and blue color value which defines the intensity of each individual color component. Colors are specified as a group; i.e., (<red>, <green>, <blue>). The range of color specification is (0, 0, 0) [for black] through (3, 3, 3) [for bright white]. Bright red has a color value of (3, 0, 0); bright green has a color value of (0, 3, 0); and bright blue has a color value of (0, 0, 3). This coding scheme provides 64 different colors.

8.9 Service Synchronization

For the most part, caption providers insert caption commands and text into the DTVCC Caption Channel stream in a real-time manner. That is, captions are transmitted shortly before they are to be displayed. This is the technique used in NTSC captioning.

DTVCC provides for an additional synchronization capability by allowing DTVCC data to be pre-sent and processed at a later time, all under the control of the decoder. The **Delay** command provides this added functionality.

8.9.1 Delay Command

Decoders shall maintain a Service Input Buffer for each of the services which can be processed simultaneously. This input buffer has a minimum size of 128 bytes. All DTVCC data for a service pass through the Service Input Buffer. Most of the time, the data falls through the buffer instantaneously, and the data are processed by the DTVCC decoder as they are received.

The **Delay** command is used to instruct the decoder to suspend processing of the Service Input Buffer data for a specified time period. This command has a time-out period specified as its parameter. This period is specified in tenths of seconds.

When a **Delay** command is encountered, the decoder waits for the specified delay time to expire before processing any other service data. During the delay interval, incoming data for the active service is buffered in the Service Input Buffer. When the delay interval expires, interpretation of the incoming data is resumed.

The **Delay** command may be used in instances where the caption provider knows that it is about to lose access to the DTVCC encoder-to-decoder stream. Prior to losing access, the caption provider pre-sends a set of DTVCC commands with one or more embedded **Delay** commands. When the caption provider loses access, the decoders process the sequencing of the buffered data according to the providers' instructions.

Any active delay interval is automatically canceled when the Service Input Buffer becomes full. The decoder begins interpreting buffered data in order not to lose any incoming data.

During the delay, the decoder should store DTVCC data for that service in a Service Input Buffer. This FIFO buffer should be able to store at least 128 bytes of DTVCC data. If the buffer becomes full, then the Delay command should be canceled, and the DTVCC data should be processed.

It is recommended that the Service Input Buffer be considered full by all decoders when it contains 128 bytes or more of DTVCC data.

The encoder should consider that a practical decoder's Service Input Buffer may need to be larger than 128 bytes since the DTVCC data arrives in units of Service Blocks. For example, if the Service Input Buffer contains 127 bytes during the execution of a Delay command, and a new Service Block arrives with 31 bytes of DTVCC data. The Service Input Buffer now contains 158 bytes of data to decode. Also, since decoding may not be instantaneous, additional Service Blocks may arrive before the buffer is emptied.

8.9.2 DelayCancel Command

Caption providers may override an active **Delay** command via the **DelayCancel** command. The **DelayCancel** command terminates any currently pending Delay command processing, and causes the decoder to begin processing any data in the Service Input Buffer.

The **DelayCancel** command shall be detected (recognized) prior to the input of Service Input Buffer. That is, the **DelayCommand** is not buffered. If it were, it would not be processed until the delay interval expires.

The **Delay** and **DelayCancel** commands are analogous to Service Input Buffer processing "suspend" and "resume". These two commands allow caption providers to preload a set of commands and with a **Delay** command (suspend) to delay their output (using the maximum delay interval), and then issue a **DelayCancel** command (resume) at a desired instance to have the commands executed en mass.

8.9.3 Reset Command

The **Reset** command reinitializes a DTVCC service. The effects of re-initialization are described in Section 8.9.5. The **Reset** command is encoded by a caption provider to reset caption service processing within decoders.

It is recommended that a **Reset** command be issued at the beginning of a captioned program, or program segment. This ensures that decoders are initialized and ready for a new program, and erases any left-over windows and captions that were not deleted as a result of such events as video up-cutting during transitions from one program source to another.

8.9.4 Reset and DelayCancel Command Recognition

In order for the **DelayCancel** and service **Reset** commands to be effective, they shall be recognized after they are retrieved from the Caption Channel Data Stream and prior to their routing to (insertion into) the Service Input Buffer. All other commands are interpreted when they are pulled from the Service Input Buffer.

Figure 21 shows this point of "pre-processing" for a decoder which simultaneously processes multiple service streams.

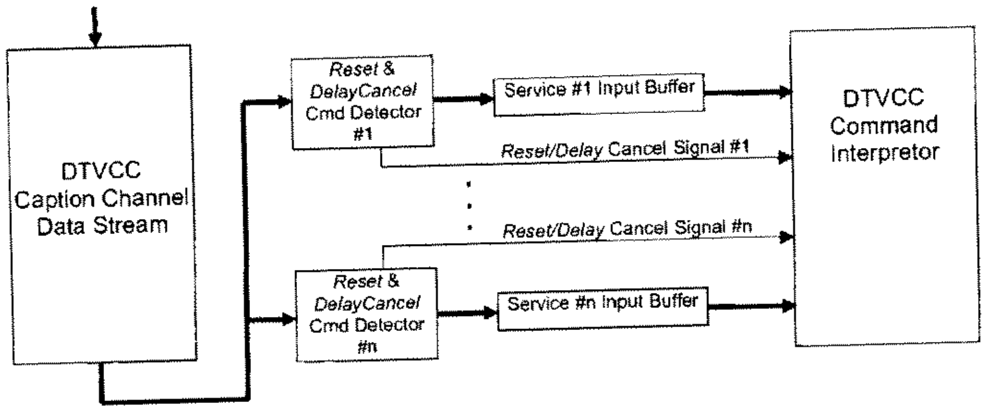
DTVCC Caption Channel Data Stream
Reset & DelayCancel Cmd Detector #1
Service #1 Input Buffer
Reset/Delay Cancel Signal #1
Reset/Delay Cancel Signal #n
Reset & DelayCancel Cmd Detector #n
Service #n Input Buffer
DTVCC Command Interpretor

Figure 21 Reset & DelayCancel Command Detector(s) and Service Input Buffers

For descriptive purposes, Figure 22 shows a conceptualization (not an actual circuit diagram) for the "*Reset & DelayCancel* Command Detector(s)"; a software implementation should be straight-forward. The Detectors may be implemented as appropriately triggered 8-bit comparators. Since each of these commands is only a single byte in length (**Reset** = 0x8F, **DelayCancel** = 0x8E), this comparison is conceptually simple, though not implementable exactly as is shown in Figure 22.

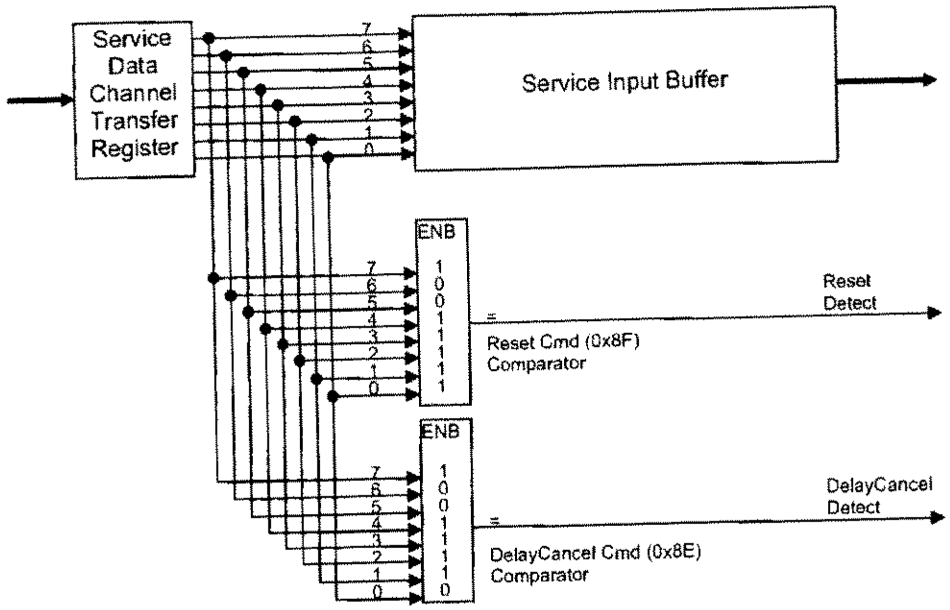

Figure 22 Reset & DelayCancel Command Detector(s) Detail

Decoders should act on Reset and DelayCancel commands when the **user_data()** containing those commands is parsed (in the "pre-processing"). Parsing **user_data()** should not stop when a Delay command is received, but parsed codes and commands should be stored in a Service Input Buffer, to be interpreted and acted upon when the Delay is complete. Reset and DelayCancel only affect the caption service in which they are sent.

Multiple caption services can be decoded concurrently, although there is no requirement to do so.

The Reset command should empty the Service Input Buffer, as described in Section 8.9.5.

8.9.5 Service Reset Conditions

The "resetting" or "re-initialization" of a service means:

a) that the service's windows are removed from the display,
b) all defined windows for the service are deleted,
c) all window and pen attributes for the service are deleted, and
d) the Service Input Buffer is cleared.

A service shall be reset when any one of the following events occur:

a) a **Reset** command is received for a service.
b) a channel change occurs.
c) the Service Input Buffer overflows.
d) A loss of continuity in the Caption Channel Packet sequence_number.

8.10 DTVCC Command Set

This section presents the commands which may be encoded by caption providers. The commands are grouped into the following command types: Window Commands, Pen Commands, Caption Text Commands, and Synchronization Commands.

8.10.1 Window Commands

These commands create, delete, modify, and display windows, and specify the current caption window for a caption service.

Command Code	Command Name	Parameters
CW0, ... , CW7	**SetCurrentWindow**	window ID
CLW	**ClearWindows**	**window map**
DF0, ... , DF7	**DefineWindow**	window ID, priority, anchor point, relative positioning, anchor vertical, anchor horizontal, row count, column count, row lock, column lock, visible, window style ID, pen style ID
DLW	**DeleteWindows**	window map
DSW	**DisplayWindows**	window map
HDW	**HideWindows**	window map
TGW	**ToggleWindows**	window map
SWA	**SetWindowAttributes**	justify, print direction, scroll direction, wordwrap, display effect, effect direction, effect speed, fill color, fill opacity, border type, border color

8.10.2 Pen Commands

These commands define pen attributes and colors.

Command Code	Command Name	Parameters
SPA	**SetPenAttributes**	pen size, font, text tag, offset, italics, underline, edge type
SPC	**SetPenColor**	fg color, fg opacity, bg color, bg opacity, edge color
SPL	**SetPenLocation**	row, column

8.10.3 Synchronization Commands

These commands control the rate of service data interpretation.

Command Code	Command Name	Parameters
DLY	**Delay**	tenths of seconds
DLC	**DelayCancel**	
RST	**Reset**	

8.10.4 Caption Text

There is no specific Text Write caption command. That is, any characters or codes from the G0, G1, G2, or G3 code sets in Service Blocks which are not part of any caption command are considered text to be written to the current window. Any such encountered text is written to the current window starting at the window's current cursor position. The window's current cursor is adjusted automatically to the row and column following the last character in the text string.

In order to assist decoders in determining the end of caption text segments for dangling segments at the end of a series of caption commands and text, an **ETX** code (0x03) from the C0 Code Space is to be inserted at the end of the text segment to terminate the segment. Text which is immediately followed by a caption command code (from the C1 Code Space) does not require the ETX code. That is, the decoder determines the end of a text segment when it encounters a caption command code, or an ETX code.

Without the **ETX** assist, decoding software may find it difficult to know if it should wait until the next service block to see if there is more caption text for a caption row when a service block terminates in text. If the last used byte in the block is an **ETX**, then the decoder knows there is no more text for this segment in a subsequent service block, and the decoder can process it as a single entity (this is especially helpful when doing Center justification of a row and word wrap processing, for example). If the last used byte in the block is a text character, then the decoder may wait until the next service block to see if there is any more text.

8.10.5 Command Descriptions

Each command is described in detail within this section.

8.10.5.1 SET CURRENT WINDOW - (CWx)

Name:	**SetCurrentWindow** - Specify current window ID
Command Type:	Window
Format:	**SetCurrentWindow** (*window ID*)
Parameters:	■ *window ID* (id) is the unique window identifier (0 - 7).
Command Coding:	**CW0, ... , CW7 = 80h, ... , 87h (10000000b, ... , 10000111b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	**0**	**0**	**0**	**0**	id_2	id_1	id_0	command

NOTE—Subscripts in Command Coding in this and subsequent Command Coding entries refer to bit number, starting with lsb.

Description: **SetCurrentWindow** specifies the window to which all subsequent window style and pen/text commands are directed. The *window ID* is required to address a window which has already been created by the **DefineWindow** command. This command initializes a "current window ID" variable internal to the decoder. SetCurrentWindow directs the following commands to the specified window: **SetWindowAttributes, SetPenAttributes, SetPenColor, SetPenLocation, WriteText.**

SetCurrentWindow Example
The one byte sequence "0x84" sets the window ID to window number four.

8.10.5.2 DEFINE WINDOW - (DF0 ... DF7)

Name:	**DefineWindow** - Create window and set initial parameters
Command Type:	Window
Format:	**DefineWindow** (*window ID, priority, anchor point, relative positioning, anchor vertical, anchor horizontal, row count, column count, row lock, column lock, visible, window style ID, pen style ID)*

Parameters:

- *window ID* (id) is the unique window identifier (0 - 7)
- *priority* (p) is the window display priority (0 - 7)
- *anchor point* (ap) is the window anchor position number to use for the window's position on the screen (0 - 8).
- *relative positioning* (rp) is a flag that, when set to 1, indicates that the *anchor vertical* (av) and *anchor horizontal* (ah) coordinates specify "relative coordinates" (i.e., percentages) instead of physical screen coordinates.
- *anchor vertical* (av) is the vertical position of the window's anchor point on the viewing screen when the window is displayed (0 – 74 for 16:9 and 4:3 systems and the *relative positioning* (rp) parameter is set to 0, or 0-99 when the 'rp' parameter is set to 1).
- *anchor horizontal* (ah) is the horizontal position of the window's anchor point on the viewing screen when the window is displayed (0 - 209 for 16:9 systems, 0 - 159 for 4:3 systems and the *relative positioning* (rp) parameter is set to 0, or 0-99 when the 'rp' parameter is set to 1).
- *row count* (rc) is the number of virtual rows of text (assuming the STANDARD *pen size;* see **SetPenAttributes**) the window body will hold minus 1(0 - 11). For example, a window with rc=2 will have three virtual rows of text.
- *column count* (cc) is the number of virtual columns of text (assuming the STANDARD *pen size;* see **SetPenAttributes**) the window body will hold minus 1 (0 - 31 for 4x3 formats, and 0 - 41 for 16x9 formats). For example, a window with cc=31 will have thirty-two virtual columns of text.
- *row lock* (rl), when set to YES, fixes the absolute number of rows of caption text the window will contain. When NO, *row lock* permits a receiver to add more rows to a window if the user selects a smaller size font other than intended by the caption provider. [YES, NO] = [1, 0].
- *column lock* (cl), when set to YES, fixes the absolute number of columns of caption text the window will contain. When NO, *column lock* permits a receiver to add more columns to a window if the user selects a smaller size font other than intended by the caption provider. [YES, NO] = [1, 0].
- *visible* (v), when set to YES, causes the window to be viewed (i.e., displayed) on the screen immediately after it is created. When set to NO, the window is not displayed (i.e., hidden) after it is created. [YES, NO] = [1, 0].
- *window style ID* (ws), when non-zero, specifies 1 of 7 static preset window attribute styles to use for the window when it is created (0 - 7). When zero during a window create, the window style is automatically set to window style #1. When zero during a window update, no window attribute parameters are changed. See **SetWindowAttributes** command.

■ *pen style ID* (ps), when non-zero, specifies 1 of 7 static preset pen attribute styles to use for the window when it is created (0 - 7). When zero during a window create, the pen style is automatically set to pen style #1. When zero during a window update, no pen attribute parameters are changed. See **SetPenAttributes** command.

Command Coding: **DF0, ... , DF7 = 98h, ... , 9Fh (10011000b, ... , 10011111b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	**0**	**0**	**1**	**1**	id_2	id_1	id_0	command
0	**0**	v	rl	cl	p_2	p_1	p_0	parm1
rp	av_7	av_6	av_5	av_4	av_3	av_1	av_0	parm2
ah_7	ah_6	ah_5	ah_4	ah_3	ah_2	ah_1	ah_0	parm3
ap_3	ap_2	ap_1	ap_0	rc_3	rc_2	rc_1	rc_0	parm4
0	**0**	cc_5	cc_4	cc_3	cc_2	cc_1	cc_0	parm5
0	**0**	ws_2	ws_1	ws_0	ps_2	ps_1	ps_0	parm6

Description: **DefineWindow** creates a window for the specified window identifier (*window ID*) and initializes the window with the non-style parameters listed in the command and with the available static style presets specified with the *window style ID* and *pen style ID* parameters. If the window is not already defined when the **DefineWindow** command is received, the window is created; otherwise it is simply updated. If the window is being created, all character positions in the window are set to the window fill color and the pen location is set to (0, 0). The **DefineWindow** command also makes the defined window the current window (see **SetCurrentWindow**).

When a window is created, a specific or automatic *window style ID* is assigned in order to preload a set of known window attribute values. These attributes can be subsequently modified with the **SetWindowAttributes** command. A *pen style ID* is assigned in the same fashion. Pen style attributes can be subsequently modified with the **SetPenAttributes** command.

When a decoder receives a **DefineWindow** command for an existing window, the command is to be ignored if the command parameters are unchanged from the previous window definition. Encoders or caption providers may periodically repeat window definitions in order for receivers to acquire a service and begin decoding and displaying captions with little delay. It is suggested that all window and pen definition and attribute commands be repeated in this way.

When an existing window is being updated (e.g., resized or moved) with the **DefineWindow** command, the pen location and pen attributes are unaffected.

DefineWindow Example
The seven byte sequence ""0x9A, 0x38, 0x4A, 0xD1, 0x8B, 0x0F, 0x11" creates the following window:

a) *window ID* = 2
b) *visible* = YES

c) *row lock* = YES. Receiver may not add more rows to the window
d) *column lock* = YES. Receiver may not add more columns to the window
e) *priority* = 0, which is the highest priority, meaning this window will be on top of any overlapping windows.
f) *relative positioning* = 0, which is required for standard screen sizes.
g) *anchor vertical* = 0x4A = 74, which means that the anchor block occupies the lowest row in the anchor grid
h) *anchor horizontal* = 0xD1 = 209, which means that the anchor block occupies the rightmost column in the anchor grid.
i) *anchor point* = 8, which is the value for the lower right corner, meaning that the anchor block is the location for the last character position in the last line of the caption window.
j) *row count* = 0xB = 11, which means that there are twelve rows, numbered from 0 to 11 in the window, with zero being the topmost.
k) *column count* = 15, which means that there are 16 characters per row, with character positions numbered from 0 to 15, with 15 being the rightmost.

If the DefineWindow command parameters differ from the previous DefineWindow command for this window (all these parameters maintained in the state of the window), or if no previous DefineWindow state exists, then the following attributes will be assigned to the window being defined:

a) *window style ID* = 2, which means Justification is left, printing is left-to-right, scrolling will be bottom-to-top (as in CEA-608-C roll-up captioning), no word-wrap will occur, captions will SNAP on and off instantaneously, the fill color for empty cells in the window is TRANSPARENT, so the underlying video shows through, there is no border drawn.
b) *pen style ID* = 1, which means that the Pen Size is STANDARD, using the default font (0), without super- or subscript offset, not italicized, not underlined, edges are shown in background color, foreground color is 2,2,2 white, which might be represented by a 68 IRE video level, background is 0,0,0 black, which is the darkest color available, which might be represented by a 7.5 IRE. The background of characters is solid-- the background color covers the character rectangle.

Table 23 refers to style #1 as "NTSC Style PopUp Captions." This style has a SOLID black fill color, making the entire window black, even before text is written into it. CEA-608-C captioning hardware usually displays character cells that are transparent until written in. Although the name implies similarity, the two captioning methods produce captions that are not similar in appearance. The Predefined Window Style #2, called "PopUp Captions w/o Black Background" looks quite like CEA-608-C captions, with the exception of black boxes before and after the text (CEA-608-C Section 3.2.2). Translated captions from CEA-608-C sources may look best when Predefined Window Style #2 is used in the CEA-708-C output.

Caption service providers should consider use of Window Style ID #2 "PopUp Captions w/o Black Background" as defined in Table 24, when designing a caption to look like CEA-608-C.

All decoders should implement Predefined Window Style #2, exceeding the minimum implementation guidelines of Section 9.12.

Style ID #	Justify	Print Direction	Scroll Direction	Word Wrap	Display Effect	Effect Direction	Effect Speed	Fill Color	Fill Opacity	Border Type	Border Color	Usage
1	LEFT	LEFT -TO- RIGHT	BOTTOM -TO- TOP	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*NTSC Style PopUp Captions*
2	LEFT	LEFT -TO- RIGHT	BOTTOM -TO- TOP	NO	SNAP	n/a	n/a	n/a	TRANS-PARENT	NONE	n/a	*PopUp Captions w/o Black Background*
3	CNTR	LEFT -TO- RIGHT	BOTTOM -TO- TOP	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*NTSC Style Centered PopUp Captions*
4	LEFT	LEFT -TO- RIGHT	BOTTOM- TO- TOP	YES	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*NTSC Style RollUp Captions*
5	LEFT	LEFT -TO- RIGHT	BOTTOM- TO- TOP	YES	SNAP	n/a	n/a	n/a	TRANS-PARENT	NONE	n/a	*RollUp Captions w/o Black Background*
6	CNTR	LEFT -TO- RIGHT	BOTTOM- TO- TOP	YES	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*NTSC Style Centered RollUp Captions*
7	LEFT	TOP -TO- BOTTOM	RIGHT -TO- LEFT	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*Ticker Tape*

Table 23 Predefined Window Style ID's

Predefined Style ID	Pen Size	Font Style	Offset	Italics	Underline	Edge Type	Foregrnd Color	Foregrnd Opacity	Backgrnd Color	Backgrnd Opacity	Edge Color	Usage
1	STNDR	0	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*Default NTSC Style**
2	STNDR	1	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Mono w/ Serif*
3	STNDR	2	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Prop w/ Serif*
4	STNDR	3	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Mono w/o Serif*
5	STNDR	4	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Prop w/o Serif*
6	STNDR	3	NORMAL	NO	NO	UNIFRM	(2,2,2) White	SOLID	n/a	TRANS-PARENT	(0,0,0) Black	*Mono w/o Serif, Bordered Text, No bg*
7	STNDR	4	NORMAL	NO	NO	UNIFRM	(2,2,2) White	SOLID	n/a	TRANS-PARENT	(0,0,0) Black	*Prop. w/o Serif, Bordered Text, No bg*

* "NTSC Style" - White Text on Black Background

Table 24 Predefined Pen Style ID's

8.10.5.3 CLEAR WINDOWS - (CLW)

Name:	**ClearWindows** - Clears Text from a set of windows
Command Type:	Window
Format:	**ClearWindows** (*window map*)
Parameters:	■ window map (w) is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents a window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A bit value of 0 indicates that the associated window is unaffected by the command.
Command Coding:	**CLW = 88h (10001000b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	**0**	**0**	**0**	**1**	**0**	**0**	**0**	command
w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	parm1

Description: **ClearWindows** removes any existing text from the specified window(s) in the window map. When a window is cleared, the entire window is filled with the window fill color.

ClearWindows Example
The two byte sequence "0x88, 0x13" clears all characters from windows 0, 1 and 4, and fills these windows with the window fill color defined in the most recently interpreted DefineWindow or SetWindowAttributes command. If the windows are not currently visible, the invisible image of the window is filled.

The pen position should be moved to 0,0 the upper-left location in each cleared window, as would be done when that window's DefineWindow was first received.

8.10.5.4 DELETE WINDOWS - (DLW)

Name: **DeleteWindows** - Deletes window definitions for a set of windows

Command Type: Window

Format: **DeleteWindows** (*window map*)

Parameters:

- *window map* (w) is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents a window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A bit value of 0 indicates that the associated window is unaffected by the command.

Command Coding: **DLW = 8Ch (10001100b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	command
w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	parm1

Description: **DeleteWindows** removes all specified windows from the receiver. For example, a *window map* value of 64 (hex) deletes windows 6 (w_6), 5 (w_5), and 2 (w_2). A *window map* value of FF (hex) deletes all defined windows in the receiver. If the current window is deleted, then the decoder's current window ID is unknown and must be reinitialized with either the **SetCurrentWindow** or **DefineWindow** command.

DeleteWindows Example

The two byte sequence "0x8C, 0xFE" erases windows 1, 2, 3, 4, 5, 6, and 7 from the display. Definitions for these windows are also deleted. Window 0 is not deleted by this sequence. If any one of the deleted windows was the current window, the value of current window ID becomes "unknown/ uninitialized", and characters following this command do not go to any window until a SetCurrentWindow or DefineWindow command is received for this service.

The deleted windows should be removed from the display when the DLW command is interpreted.

8.10.5.5 DISPLAY WINDOWS - (DSW)

Name: **DisplayWindows** – Causes a set of windows to become visible

Command Type: Window

Format: **DisplayWindows** (*window map*)

Parameters: ■ *window map* (w) is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents a window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A value of 0 indicates that the associated window is unaffected by the command.

Command Coding: **DSW = 89h (10001001b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	0	1	0	0	1	command
w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	parm1

Description: **DisplayWindows** causes the specified, existing windows to be visible on the receiver display screen. For example, a *window map* value of 96 (hex) displays windows 7 (w_7), 4 (w_4), 2 (w_2), and 1 (w_1). A *window map* value of FF (hex) causes all existing windows in the receiver to be displayed. This command does not affect the current window ID.

DisplayWindows Example
The two byte sequence "0x89, 0x80" causes window 7 to be displayed (made visible) by decoders which have a window definition for that window (have received a DefineWindow command). The window will appear using the display effect defined for that window.

If the window has a TRANSPARENT border and background and no characters in it when the window's state is set to visible, no graphic is be displayed. If two windows with equal priority and overlapping locations are displayed by this command, the decoder may show them with either window on top, or preferably, move the display location of the windows to allow all text to be visible. (The overlap situation should only occur when the user preferences have changed window dimensions.)

8.10.5.6 HIDE WINDOWS - (HDW)

Name: **HideWindows –** Causes a set of windows to become invisible

Command Type: Window

Format: **HideWindows** (*window map*)

Parameters: ■ *window map* is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents the window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A value of 0 indicates that the associated window is unaffected by the command.

Command Coding: **HDW = 8Ah (10001010b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	0	1	0	1	0	command
w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	parm1

Description: **HideWindows** causes all specified and currently defined windows to be removed from the receiver display screen. For example, a *window map* value of 72 (hex) hides windows 6 (w_6), 5 (w_5), 4 (w_4), and 1 (w_1). A *window map* value of FF (hex) causes all existing windows in the receiver to be hidden.

HideWindows Example
The two byte sequence "0x8A, 0x7E" erases windows 1, 2, 3, 4, 5, and 6 from the display using the display effects defined for each of the windows. The definitions for these windows are retained, as is the value of current window ID, and the text data in each of the windows. The state of the erased windows is set to hidden (invisible).

If windows 0 or 7 were visible, they remain visible; if either was hidden by one of the other windows, it should now appear on the display.

8.10.5.7 TOGGLE WINDOWS - (TGW)

Name:	**Toggle Windows** - Toggles Display/Hide status of a set of windows
Command Type:	Window
Format:	**Toggle Windows** (*window map*)
Parameters:	■ <u>*window map*</u> is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents the window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A value of 0 indicates that the associated window is unaffected by the command.
Command Coding:	**TGW = 8Bh (10001011b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	0	1	0	1	1	command
w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	parm1

Description: **Toggle Windows** causes all specified and currently defined windows to toggle their display/hide status. That is, the specified windows in the *window map* which are currently displayed will be hidden, and the hidden ones will be displayed. For example, a *window map* value of 83 (hex) toggles windows 7 (w_7), 1 (w_1), and 0 (w_0). A *window map* value of FF (hex) causes all existing windows in the receiver to be toggled.

ToggleWindows Example
The two byte sequence "0x8B, 0xF6" erases or shows windows 1, 2, 4, 5, 6, and 7, depending on their visibility states, using the display effects defined for each. The definitions for these windows are retained, as is the value of current window ID, and the text data in each of the windows.

8.10.5.8 SET WINDOW ATTRIBUTES - (SWA)

Name:	**SetWindowAttributes** - Defines the window styles for the current window.
Command Type:	Window
Format:	**SetWindowAttributes** (*justify, print direction, scroll direction, wordwrap, display effect, effect direction, effect speed, fill color, fill opacity, border type, border color)*

Parameters:

- *justify* (j) specifies how the text to be written in the window will be justified. [LEFT, RIGHT, CENTER, FULL] == [0, 1, 2, 3].
- *print direction* (pd) specifies in which direction text will be written in the window. [LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP] == [0, 1, 2, 3].
- *scroll direction* (sd) specifies which direction text will scroll when the end of a caption "line" is reached. [LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP] == [0, 1, 2, 3]..
- *wordwrap* (ww), when set to YES, word wrapping is enabled. When set to NO, wordwrapping is disabled. [YES, NO] == [1, 0].
- *display effect* (de) specifies the effect that is to take place when the window is displayed and when it is hidden. When the SNAP effect is chosen, the window will pop-on the screen when the window is displayed and pop-off when the window is hidden. The FADE effect causes the window to fade onto and off of the screen at the specified *effect rate*. The WIPE effect causes the window to swipe onto and off of the screen at the specified *effect rate* and *effect direction*. [SNAP, FADE, WIPE] == [0, 1, 2].
- *effect direction* (ed) specifies which direction a WIPE window will appear on the screen. Note that a WIPE window will wipe-off of the screen in the opposite direction from which it wiped-on. [LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP] == [0, 1, 2, 3].
- *effect speed* (es) specifies, in .5 second units, how fast windows with WIPE and FADE effects will appear and disappear from the screen when they are displayed and hidden. The effect speed value may range from 1 to 15, approximating .5 (1 x .5) to 7.5 (15 x .5) seconds of effect speed variation.
- *fill color* (fr, fg, fb) is the color of the windows interior (see Section 8.4.11).
- *fill opacity* (fo) is the characteristic of the fill color of the window and the window border. [SOLID, FLASH, TRANSLUCENT, TRANSPARENT] == [0, 1, 2, 3].
- *border type* (bt) defines the type of outer edge surrounding the window. [NONE, RAISED, DEPRESSED, UNIFORM, SHADOW_LEFT, SHADOW_RIGHT] == [0, 1, 2, 3, 4, 5].
- *border color* (br, bg, bb) is the color of the windows outer edge (see Section 8.4.11).

Command Coding: **SWA = 97h (10010111b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	**0**	**0**	**1**	**0**	**1**	**1**	**1**	command
fo_1	fo_0	fr_1	fr_0	fg_1	fg_0	fb_1	fb_0	parm1
bt_1	bt_0	br_1	br_0	bg_1	bg_0	bb_1	bb_0	parm2
bt_2	ww	pd_1	pd_0	sd_1	sd_0	j_1	j_0	parm3
es_3	es_2	es_1	es_0	ed_1	ed_0	de_1	de_0	parm4

Description: **SetWindowAttributes** assigns the specified style attributes to the current window. This command can be issued any number of times to an existing window. The style attributes will overwrite any existing attributes assigned to the window.

SetWindowAttributes Example

The five byte command "0x97, 0x64, 0x53, 0x88, 0x22" modifies the attributes of the current window (as specified by current window ID). It immediately changes the following attributes of the window:

a) *fill color* = (fill red = 2, fill green = 1, fill blue = 0), which is a dull orange.
b) *fill opacity* = 1, which means that the window fill and border FLASH. This does not effect the foreground or backgrounds of text in the window, nor does it effect transparent spaces.
c) *border color* = (border red = 1, border green = 0, border blue = 3), which is a violet-purple.
d) *border type* = 5 = SHADOW_RIGHT, which can be represented as a dark blue square, offset slightly to the lower right, overdrawn with a violet-purple square.
e) *wordwrap* = NO, which means that any display image reformatting of the window's data is removed, and the text displayed at the positions at which they were received. The operation of wordwrap is not currently defined.
f) *justify* = LEFT, which means that all text in the window are left justified.

The remaining values in the SetWindowAttributes command do not effect the display of the window, but do change the way the window acts:

a) *print direction* = 0 = LEFT_TO_RIGHT, which means that when a character is placed at the current cursor position, the character insertion location should move one character higher in the current row. (If print direction was RIGHT_TO_LEFT, the character insertion location would move one character lower in the current row, and if it were TOP_TO_BOTTOM, it would move to a position in the same column, one row higher.
b) *scroll direction* = 2 = TOP_TO_BOTTOM, which defines the cursor movement per [[Need Figure 16]]. When a carriage return is encountered, the cursor is moved to column=0 of the row above the current row, and if the current row is row 0, the text in the window is scrolled down by one row.
c) *display effect* = 2 = WIPE, which means that the appearance and disappearance of windows take the form of a "key wipe", as if the text was painted on from one side to the other.
d) *effect direction* = 0 = LEFT_TO_RIGHT, which means that when the window is set to visible, it appears on the left side first, then progressively more of the window appears until the right edge appears. When the window is set to hidden, the effect direction is the same, the window disappears on the left side first.
e) *effect speed* = 2 = 1.0 seconds. The wipe effect takes one second to go from the left to the right.

CEA-708-C addresses whether the speed (actually, the duration) should be applied across the entire viewing area, or across the window. A four character wide window may take the same time to be removed as a twenty character wide window, or it may take one fifth the time. Further, the wipe may begin at one side of the screen, or at one side of each window.

8.10.5.9 SET PEN ATTRIBUTES - (SPA)

Name:	**SetPenAttributes** - Assign pen style attributes for the current window.
Command Type:	Pen
Format:	**SetPenAttributes** (pen size, font, text tag, offset, italics, underline, edge type)

Parameters:

■ *pen size* (s) defines which of three pen sizes is to be used for text written to the current window, as specified by the current window ID. Note that the pen size displayed on the screen can be overridden by the user. [SMALL, STANDARD, LARGE] == [0, 1, 2].

■ *font style* (fs) specifies which one of 8 different predefined font styles (see Section 8.5.3) to be used for text written to the current window (0 - 7).

0 - Default (undefined)
1 - Monospaced with serifs
2 - Proportionally spaced with serifs
3 - Monospaced without serifs
4 - Proportionally spaced without serifs
5 - Casual font type
6 - Cursive font type
7 - Small capitals

■ *text tag* (tt) specifies which one of 16 different predefined caption text function tags (see Section 8.5.9) is to be associated with the following caption text to be written to the current window (0 - 15).

0 - Dialog
1 - Source or speaker ID
2 - Electronically reproduced voice
3 - Dialog in language other than primary
4 - Voiceover
5 - Audible Translation
6 - Subtitle Translation
7 - Voice quality description
8 - Song Lyrics
9 - Sound effect description
10 - Musical score description
11 - Expletive
12 to 14 - (undefined)
15 - Text not to be displayed

■ *offset* (o) specifies sub-scripting and super-scripting attributes for text written to the current window. [SUBSCRIPT, NORMAL, SUPERSCRIPT] == [0, 1, 2].

■ *italics* (i) specifies if text written to the current window is italicized. [YES, NO] == [1, 0].

■ *underline* (u) specifies if text written to the current window is underlined. [YES, NO] == [1, 0].

■ *edge type* (et) is the type of outlined edge of the text. [NONE, RAISED, DEPRESSED, UNIFORM, LEFT_DROP_SHADOW, RIGHT_DROP_SHADOW] == [0, 1, 2, 3, 4, 5].

Command Coding: **SPA = 90h (10010000b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	**0**	**0**	**1**	**0**	**0**	**0**	**0**	command
tt_3	tt_2	tt_1	tt_0	o_1	o_0	s_1	s_0	parm1
i	u	et_2	et_1	et_0	fs_2	fs_1	fs_0	

Description: **SetPenAttributes** assigns pen style attributes for the currently defined window. Text written to the current window will have the attributes specified by the most recent **SetPenAttributes** command written to the window. Pen attributes for a window can be changed as often as desired. These attributes will remain in effect for the window during its entire existence.

SetPenAttributes Example
The three byte command "0x90, 0x4A, 0xCA" modifies the pen used for subsequent writing in the current window.

a) *text tag* = 4 = Voiceover, which has been chosen by the decoder designer or user to indicate that the audio is an off-camera voiceover. Because *text tag ≠ 0*, the text tag takes priority over the other attributes in SetPenAttributes, and they are ignored.

If the text tag had been zero (Dialog), the following values would have been used:

a) *pen size* = 0 = SMALL, which sets the font size to small unless it is overridden by user command.
b) *font style* = 4 = Proportionally spaced without serifs, unless overridden by user command.
c) *offset* = 2 = SUPERSCRIPT, meaning that the text would be raised relative to normal.
d) *italics* = 1, meaning that characters should appear in the italic form of the chosen font.
e) *underline* = 1, meaning that subsequent characters appear underlined.
f) *edge type* = 1 = RAISED, meaning that subsequent characters appear with a raised edge.

8.10.5.10 SET PEN COLOR - (SPC)

Name: **SetPenColor** - Assign styles to a dynamic preset style number.

Command Type: Pen

Format: **Set Pen Color** (*fg color, fg opacity, bg color, bg opacity, edge color)*

Parameters:

- *fg color* (fr, fg, fb) is the color of the text foreground body (see Section 8.5.6).
- *fg opacity* (fo) is the characteristic of the text foreground body. [SOLID, FLASH, TRANSLUCENT, TRANSPARENT] == [0, 1, 2, 3].
- *bg color* (br, bg, bb) is the color of the background box surrounding the window text (see Section 8.5.7).
- *bg opacity* (bo) is the characteristic of the text background. [SOLID, FLASH, TRANSLUCENT, TRANSPARENT] == [0, 1, 2, 3].
- *edge color* (er, eg, eb) is the color of the outlined edges of the text. The text character edges have the same opacity value as *fg opacity* (see Section 8.5.6).

Command Coding: **SPC = 91h (10010001b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	1	0	0	0	1	command
fo_1	fo_0	fr_1	fr_0	fg_1	fg_0	fb_1	fb_0	parm1
bo_1	bo_0	br_1	br_0	bg_1	bg_0	bb_1	bb_0	parm2
0	0	er_1	er_0	eg_1	eg_0	eb_1	eb_0	parm3

Description: **SetPenColor** assigns the pen color attributes for the current window, as specified by the current window ID. Text written to the current window will have the color attributes specified by the most recent **SetPenColor** command written to the window. Pen color attributes for a window can be changed as often as desired. These attributes will remain in effect for the window during its entire existence.

SetPenColor Example
The four byte command "0x91,0x3F,0xC0,0x15" modifies the color of the pen used for subsequent writing in the current window.

a) *fg color* = (red = 3, green = 3, blue = 3) = BRIGHT WHITE
b) *fg opacity* = 0 = SOLID.
c) *bg color* = (red = 0, green = 0, blue = 0) = BLACK
d) *bg opacity* = 3 = TRANSPARENT, meaning that the window's *fill color* is used as the background color, or, for example, if it is TRANSPARENT, the video shows through.
e) *edge color* = (red = 1, green =1, blue = 1) = dark gray.

If SetPenAttributes has defined a pen with edges, the color of the edge for those characters is be dark gray. If the foreground was TRANSPARENT, the edge would be transparent as well (when *edge type ≠ NONE*).

8.10.5.11 SET PEN LOCATION - (SPL)

Name: **SetPenLocation - Specifies the pen cursor location within a window.**

Command Type: Pen

Format: **SetPenLocation** *(row, column)*

Parameters:
- *row* (r) is the text row within the current window's text buffer (0-14).
- *column* (c) is the text column within the current window's text buffer (0-31 for 4:3 formats, 0-41 for 16:9 formats).

Command Coding: **SPL = 92h (10010010b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	1	0	0	1	0	command
0	0	0	0	r_3	r_2	r_1	r_0	parm1
0	0	c_5	c_4	c_3	c_2	c_1	c_0	parm2

Description: **SetPenLocation** repositions the pen cursor for the current window, as specified by the current window ID. When the window justification type is "left," The next group of text written to the current window will start at the specified row and column, and justification will be ignored. When the window justification type is not left and the print direction is left-to-right or right-to-left, the column parameter shall be ignored. When the window justification type is not left and the print direction is top-to-bottom or bottom-to-top, the row parameter shall be ignored. When the window justification type is not left, text shall be formatted based upon the current window justification type. Note that if a window is not *locked* (see **DefineWindow**) and the SMALL *pen size* (see **SetPenAttributes**) is in effect, more than 12 rows and 36 columns could possibly be addressed.

SetPenLocation Example
The three byte command "0x92, 0x05, 0x08" modifies the position of the next character placed into the window.

a) *row* = 5, means that the pen location is in the sixth row of the window (rows zero is the topmost).
b) *column* = 8, means that the pen location is in the ninth column of the window (column zero is the leftmost).

The location within the internal buffer is not changed by font size or proportional spacing, nor is the location within the internal buffer changed by justification. Therefore, if justification is set to CENTER, location 3,3 might not be displayed underneath location 2,3 (if the number of characters in each line differ).

The location within the internal buffer is not changed by print direction. Row = 0 always specifies the top row of the window, and column =0 is always the leftmost column.

8.10.5.12 DELAY - (DLY)

Name:	**Delay -** Delays service data interpretation
Command Type:	Synchronization
Format:	**Delay** (*tenths of seconds)*
Parameters:	■ tenths of seconds (t) is the number of tenths of seconds to delay before recommencing service data interpretation.
Command Coding:	**DLY = 8Dh (10001101b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	0	1	1	0	1	command
t_7	t_6	t_5	t_4	t_3	t_2	t_1	t_0	parm1

Description:

Delay instructs receivers to suspend interpretation of the current service's command input buffer. The delay is specified in tenths of seconds. Once the delay time expires, interpretation of caption commands recommences.

The delay value may range from 1 to 255 -- which specifies an effective delay time from 1/10 to 25.5 (255/10) seconds.

A delay for a service remains in effect until one of the following occurs:

- the specified delay time expires
- a DelayCancel command is received
- the service's input buffer becomes full
- a service Reset command is received

Delay Example
The two byte command "0x8D, 0x0B" suspends interpretation of the current service's command buffer for 1.1 seconds. Other caption services, and CEA-608-C caption services are not affected.

a) *tenths of seconds* = 0xB = 11.

8.10.5.13 DELAY CANCEL - (DLC)

Name: **DelayCancel** - Cancels an Active Delay Command

Command Type: Synchronization

Format: **DelayCancel**

Parameters: none

Command Coding: **DLC = 8Eh (10001110b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	0	1	1	1	0	command

Description: **DelayCancel** command terminates any active **Delay** command processing within the decoder.

DelayCancel Example
The one byte command "0x8E" causes any active delay to be cancelled. See Section 8.9.

8.10.5.14 RESET - (RST)

Name:	**Reset** - Resets the Caption Channel Service
Command Type:	Synchronization
Format:	**Reset**
Parameters:	none
Command Coding:	**RST = 8Fh (10001111b)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	0	0	1	1	1	1	command

Description: **Reset** command reinitializes the service for which it is received.

Reset Example
This one byte command "0x8F" immediately resets the current service. Other caption services, and CEA-608-C caption services are not affected.

8.11 Proper Order of Data

CEA-608-C Section B.8 states:

> It is recommended that caption providers follow a consistent algorithm for data insertion, even though the FCC rules theoretically allow for variations. Each burst of data should be immediately preceded by a miscellaneous control code to identify the data channel, mode and/or caption style.

This "proper order of data" specification in CEA-608-C defines the proper order of data. In this manner, the caption data is authored as "bursts of data," where each burst contains one caption or one line of a caption. Each "burst" is displayed properly, independently of any caption bursts that preceded it. When the viewer changes channels, proper captions are displayed as soon as the next complete burst is received.

It is recommended that caption providers follow a consistent algorithm when authoring CEA-708-C captions. Caption providers should author captions as a series of independent captions or "bursts of caption data." Since a viewer may tune to a program at any time, each caption should contain all commands necessary to display captions properly, without depending on any previous caption data. Where appropriate, the Reset command should be included in the captioning sequence. The Reset command is a simple method of placing decoders into a known state. The following recommended sequences illustrate the proper order of data for simple captioning styles, and provide guidance for more complex captioning styles.

8.11.1.1 Simple Roll-up Style Captions

For each simple roll-up style caption, the following sequence is recommended:

a) DeleteWindows – delete all windows except for the one being used
b) DefineWindow – define/redefine the window being used
c) SetWindowAttributes – optional (customize the window being used)
d) SetPenLocation – to leftmost column of bottom row of window
e) CR (carriage return character: 0x0D)
f) Pen Commands & Caption Text

It is equally valid for the CR to come immediately before the SetPenLocation command instead of immediately after.

If the SetPenLocation command is sent before the CR (as given in the recommended sequence), then all lines in the caption scroll into the window as they are sent.

If the CR is sent before the SetPenLocation command, then the first line of the caption (after the window is first defined or the receiver begins receiving the data) displays at the bottom of the window without scrolling. In addition, the rest of the lines should scroll into the window as they are sent, although how the receiver would handle the SetPenLocation command while it is performing a scroll is questioned.

8.11.1.2 Simple Paint-on Style Captions

For each simple paint-on style caption, the following sequence is recommended:

a) Reset – delete all windows
b) DefineWindow – define the window being used
c) SetWindowAttributes – optional (customize the window being used)
d) Pen Commands & Caption Text

If other windows are to persist when the new paint-on window is being prepared, Reset may be replaced by DeleteWindows command removing unused windows, and ClearWindows may be inserted before Pen Commands to clear the redefined window.

8.11.1.3 Simple Pop-on Style Captions
For simple pop-on style captions, two windows are used. For each simple pop-on style caption, the following sequence is recommended:

a) DeleteWindows – delete all windows except for the one being displayed (visible)
b) DefineWindow – define the hidden (not visible) window
c) SetWindowAttributes – optional (customize the hidden window)
d) Pen Commands & Caption Text
e) ClearWindows – clear the displayed (visible) window
f) ToggleWindows – toggle the hidden/displayed status for both windows being used

The ClearWindows command is intended to create a brief "blink" between captions, an effect which is often desirable. It may be necessary to insert a Delay command between the ClearWindows and ToggleWindows commands to achieve the desired blink. The ClearWindows command can be omitted when no blink is desired (each caption will appear instantly to replace another with no blink).

9 DTVCC Decoder Manufacturer Recommendations
The following are the recommendations for DTV Closed Captioning decoder implementation including DTV, SDTV and HDTV. These recommendations are directed to the least common denominator of all of the DTVCC features described in the previous sections. Although voluntary, these recommendations should be considered as requirements for a realistic minimal implementation of the DTVCC capabilities. These minimal recommendations provide a bridge from NTSC (CEA-608-C) captioning implementation to the eventual full-feature implementation of CEA-708-C.

It should be emphasized that these minimum recommendations are not intended to, and should not, restrict caption providers from using the whole suite of DTVCC commands and their extensive capabilities. The following sections address the minimum recommendations that have been anticipated, but may not cover all conditions and manifestations. It is up to the manufacturer to consider all situations that are not explicitly presented herein.

> NOTE—While every effort has been made to exactly match the decoder capabilities which are required by the FCC in this standard, implementors should examine 47 C.F.R. §15.119 for products destined for the United States of America as that takes precedence for such products.

9.1 DTVCC Section 4.2 - Pre-Allocated Bandwidth
While the DTVCC Caption Channel provides a continuous 9600 bps bitstream within the DTVCC Transport Channel, the individual bandwidth allocated to any single service shall not exceed 25% of the total bandwidth averaged over any 1 second time interval. This limit permits a maximum, average captioning data rate of 2400 bps per service.

That is, decoders need only implement enough buffering and processing power to handle a maximum of 2400 bps for each service. In effect, when this limit is exceeded for a service, the input storage buffer allocated for the service may overflow, resulting in data loss.

> NOTE—In contrast, the per-service limitation addressed above still provides a five-fold enhancement over the maximum possible NTSC Closed-Caption service data rate of 60 bps.

For each MPEG picture, sum the number of Packet Data bytes used in all frames with display times within the previous second (including the current picture). The total number of bytes should not be greater than (2400/8)=300. This method measures data rate in display order, not transmission order. Because this measurement method is only a recommendation, encoders may calculate the bit rate in other ways, including averaging over the previous hour. Because of this lack of a standard, it is further recommended that decoder designs have the ability to process the entire 9600 bps bandwidth.

9.2 DTVCC Section 6.1 - Services
Decoders should be capable of decoding and processing data for at least one (1) service. Decoders shall be capable of decoding and processing the Caption Service Directory data.

9.3 DTVCC Section 6.2 - Service Blocks

Decoders shall be capable of decoding all Caption Channel Block Headers consisting of Standard Service Headers, Extended Service Block Headers, and Null Block headers. However, decoding of the data is required only for Standard Service Blocks (Service IDs <= 6), and then only if the characters for the corresponding language are supported. See 47 C.F.R. §15.122.

FCC regulations, 47 C.F.R. 15.122 c)2), require decoders to be able to display the directory for services 1 through 6. Service decoding and directory display for services numbered 7 or greater are optional.

Decoders should be capable of decoding any of the services described in the **caption_service_descriptor()** as currently defined in ATSC A/65C.

9.3.1 caption_service_descriptor() and DTVCC Services

The definition of the **caption_service_descriptor()** in ATSC A/65C includes several limitations, not found in CEA-708-C.

Further, the **caption_service_descriptor()** only allows 16 total services, where CEA-708-C allows 63. Sending 63 caption services is not prohibited, even if only 16 are described in **caption_service_descriptor()**.

In addition to the required support for 6 standard DTV CC services, processing (and forwarding on NTSC outputs-see Section 9.23) of the embedded CEA-608 service or services shall be supported.

Decoders may use other techniques to determine what caption services are in use. Unfortunately, a caption service may only be present in the stream for a few frames per hour, so detecting services based on the occurrence of service blocks is not reliable or responsive, and should not be used as the only method of generating a caption service list.

9.3.2 Decoding 16 Services in caption_service_descriptor()

While CEA-708-C does not require decoding of caption services 7-63, decoder manufacturers should not limit their decoder to the six basic services, but should include all that are available.

9.3.3 Selecting NTSC Services Regardless of Presence of caption_service_descriptor()

Decoders that allow selection of NTSC caption services (as a fall back when no DTVCC are present) should allow any one of the four caption channels (CC1-CC4) to be selected, even if no **caption_service_descriptor()** is present. This allows decoders to continue access to CC2 and CC4.

9.3.4 Handling Multiple Entries Indicating the Same Service

Decoders should be designed to receive descriptors with more than one entry with the same value for **line21_field**. Multiple entries for sets of data can exist following each language code. Each entry with **digital_cc** false (set to '0') should be interpreted as containing CEA-608-C datastream bytes. Some early implementations may contain entries in the "for loop," which describe NTSC caption services using duplicate values for **line21_field,** and all but the first such in each set for a language code should be ignored.

9.3.5 Ignoring Reserved Field in caption_service_descriptor()

Decoders should tolerate any values and not interpret the 5 bit 'reserved' field in the **caption_service_descriptor()**. This may be used in the future.

9.3.5.1 Allowing Service Selection by Number

Decoders should include an override to set the caption decoder to a particular service number, whether consistent descriptors are present or not.

This allows for future expansion of the **caption_service_descriptor()** without denying access to decoders designed to current CEA-708-C and ATSC A/65C standards.

9.4 DTVCC Section 7.1 - Code Space Organization

Decoders must support Code Space C0, G0, C1, and G1 in their entirety.

The following characters within code space G2 must be supported:

a) transparent space ($_{TSP}$)
b) non-breaking transparent space ($_{NBTSP}$)
c) solid block (█)
d) trademark symbol (™)
e) Latin-1 characters (Š, Œ, š, œ, Ÿ)

The substitutions in Table 25 shall be made if a decoder does not support the specific G2 characters shown.

G2 Character	Substitute With
open single quote ('), G2 char code 0x31	G0 single quote ('), char code 0x27
close single quote('), G2 char code 0x32	G0 single quote ('), char code 0x27
open double quote ("), G2 char code 0x33	G0 double quote ("), char code 0x22
close double quote ("), G2 char code 0x34	G0 double quote ("), char code 0x22
bold bullet (•), G2 char code 0x35	G1 bullet (·), char code 0xB7
ellipsis(…), G2 char code 0x25	G0 underscore (_), char code 0x5F
one-eighth (⅛), G2 char code 0x76	G0 percent sign (%), char code 0x25
three-eighths (⅜), G2 char code 0x77	G0 percent sign (%), char code 0x25
five-eighths (⅝), G2 char code 0x78	G0 percent sign (%), char code 0x25
seven-eighths (⅞), G2 char code 0x79	G0 percent sign (%), char code 0x25
vertical border (│), G2 char code 0x7A	G0 stroke (|), char code 0x7C
upper-right border (┐), G2 char code 0x7B	G0 dash (-), char code 0x2D
lower-left border (└), G2 char code 0x7C	G0 dash (-), char code 0x2D
horizontal border (—), G2 char code 0x7D	G0 dash (-), char code 0x2D
lower-right border (┘), G2 char code 0x7E	G0 dash (-), char code 0x2D
upper-left border (┌), G2 char code 0x7F	G0 dash (-), char code 0x2D

Table 25 G2 Character Substitution Table

Support for code spaces C2, C3, and G3 is optional.

All unsupported graphic symbols in the G3 code space shall be substituted with the G0 underscore character (_), char code 0x5F.

9.5 DTVCC Section 8.2 - Screen Coordinates

Table 26 specifies the screen coordinate resolutions and limits for anchor point positioning in 4:3 and 16:9 display formats, and the number of characters per row.

Screen Aspect Ratio	Maximum Anchor Position Resolution	Minimum Anchor Position Resolution	Maximum Displayed Rows	Maximum Characters per Row
4:3	75v x 160h	15v x 32h	4	32
16:9	75v x 210h	15v x 42h	4	42
other	75v x (5 x H)	15v x H*	4	*

Table 26 Screen Coordinate Resolutions & Limits

*H = 32 x (the width of the screen in relation to a 4:3 display). For example, the 16:9 format is 1/3 wider than a 4:3 display; thus, H = 32 * 4/3 = 42.667, or 42.

This means that the minimum grid resolution for a 4:3 aspect ratio device is 15 vertical positions x 32 horizontal positions. This minimum grid resolution for 16:9 ratio device is 15 vertical positions x 42 horizontal positions. These minimum grid sizes are to cover the entire safe-title area of the corresponding screen.

The minimum coordinates equate to a 1/5 reduction in the maximum horizontal and vertical grid resolution coordinates. Caption providers are to use the maximum coordinate system values when specifying anchor point positions. Decoders using the minimum resolution are to divide the provided horizontal and vertical screen coordinates by 5 to derive the equivalent minimum coordinates.

Figure 23 shows a reduced resolution grid, using 10 of the 15 vertical "positions" and 5 of the 32 horizontal "positions". These can be understood as "super cells", with five times the height and width of regular cells. Windows then have one of these "super cells" as their corner, center or side. The "location" in Figure 23 is the 5x5 "super cell". This cell will be entirely within the window.

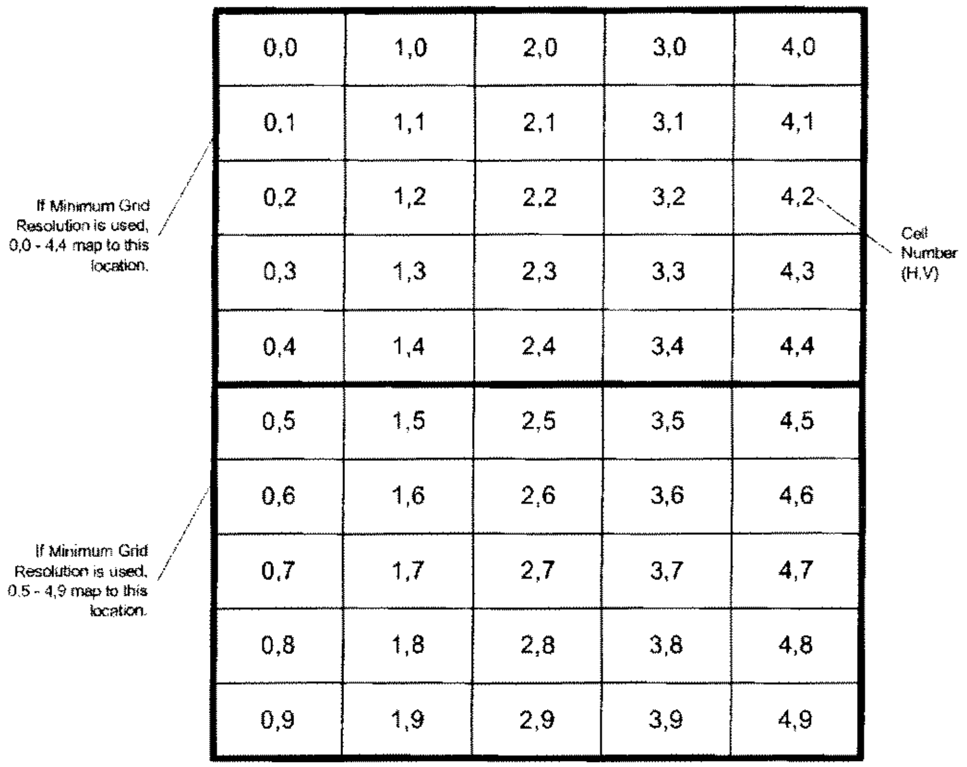

Figure 23 Minimum Grid Location Super Cell Example

See Section 8.2 for numbering of grid for window location.

Any caption targeted for both 4:3 and 16:9 devices is limited to 32 contiguous characters per row. If a caption is received by a 4:3 device that is targeted for a 16:9 display only, or requires a window width greater than 32 characters, then the caption may be completely disregarded by the decoder. 16:9 devices should be able to process and display captions intended for 4:3 displays, providing all other minimum recommendations are met.

If the resulting size of any window is larger than the safe title area for the corresponding display's aspect ratio, then this window will be completely disregarded.

If a change cannot be made without exceeding the safe title area, the decoder may change any pen attribute of any characters in the window to make the window fit.

Visibility of the caption message is the primary goal of closed captioning. Fonts, emphasis and special effects are secondary to information display. For this reason, making the text visible should take priority.

This recommendation could cause LARGE characters to revert to STANDARD, italic to non-italic, or other font changes. It should only be used as a last resort. It is known that use of STANDARD, monospaced font for all characters produces a window that fits within the safe title area. If no other methods of providing a visible caption provide an understandable caption display, this recommendation can provide a solution.

> NOTE—There are some situations with no solution, such as use of 32 LARGE characters on a 4:3 display.

9.6 DTVCC Section 8.4 - Caption Windows

Minimum decoder implementations shall be capable of displaying at least 4 windows at one time. If more than 4 windows are defined in the caption stream, the decoder may disregard the youngest and lowest priority window definition(s). Caption providers should be aware window numbers 5 and higher might not be displayed by some receivers.

Minimum decoder implementations shall maintain storage to support at least 8 rows of captions. This storage is needed for the worst-case support of a displayed window containing 4 rows of captioning and a non-displayed window which is buffering the incoming rows for the next 4-row caption.

This discussion is not intended to preclude the possibility that caption windows may be defined with more than four rows (rc>3). Decoders should not ignore caption windows defined with rc up to 11, as in Section 8.10.5.2, although only four rows of text need be displayed on the screen. Caption providers should consider, however, that some legacy decoder implementations ignore caption windows with rc>3.

9.7 DTVCC Section 8.4.2 - Window Priority

Decoders do not need to support overlapped windows. If a window overlaps another window, the overlapped window need not be displayed by the decoder. Decoders may support overlapped windows as an option.

Caption providers should use extra caution when authoring captions employing multiple displayed windows. Due to ambiguous specifications of character row height versus window anchor location definition, decoder implementations of window positioning and handling of multiple windows may vary. Windows that are intended to be close but not overlapping may be interpreted to be overlapping in some decoder implementations, in which case they might not be displayed on such decoders. Furthermore, if the viewer overrides the default character size with a larger size, the windows may be deemed to be overlapping, and similarly not be displayed.

9.8 DTVCC Section 8.4.6 - Window Size

At a minimum, decoders will assume that all windows have rows and columns "locked". This implies that if a decoder implements the optional SMALL pen-size, then un-wrapping words, when shrinking captions, need not be implemented. Also, if a decoder implements the optional LARGE pen size, then word wrapping (when enlarging captions) need not be implemented.

Caption providers should not set the "wordwrap" parameter in the SWA command, and should always set the row lock and column lock parameters in the DefineWindow command.

9.9 DTVCC Section 8.4.8 - Word Wrapping

Decoders may support word wrapping as an option.

9.10 DTVCC Section 8.4.9 - Window Text Painting

9.10.1 Justification

All decoders should implement "left", "right", and "center" caption-text justification. Implementation of "full" justification is optional. If "full" justification is not implemented, fully justified captions should be treated as though they are "left" justified. See Section 8.4.9.1.2.

For "left" justification, decoders should display any portion of a received row of text when it is received. For "center", "right", and "full" justification, decoders may display any portion of a received row of text when it is received, or may delay display of a received row of text until reception of a row completion indicator. A row completion indicator is defined as receipt of a CR, ETX or any other command, except SetPenColor, SetPenAttributes, or SetPenLocation where the pen relocation is within the same row.

Receipt of a character for a displayed row which already contains text with "center", "right" or "full" justification will cause the row to be cleared prior to the display of the newly received character and any subsequent characters. Receipt of a justification command which changes the last received justification for a given window will cause the window to be cleared.

It is recommended that any text received prior to the new SWA command not be redisplayed, and only text received subsequent to the new SWA command be written with the new alignment. Caption authors should not rely on this operation, but should issue a ClearWindows, Delete Windows or Reset command to empty the window being redefined.

While literal interpretation could cause every character received to clear the current row of a justified window, clearing should only apply when text is in a window and a "row completion indicator" has been received. Because they are not characters to be displayed, reception of CR, ETX and NUL codes should not cause the display to erase a justified row. Reception of FF and HCR cause the row to be erased because of the nature of the code. Other non-character codes, such as BS, NBS, TSP and NBTSP affect the current line's display, and would, therefore, cause the row to be cleared. This should apply to completed rows when the window is hidden (visible = NO) and when the window is covered over due to priority.

9.10.2 Print Direction

At a minimum, decoders shall support LEFT_TO_RIGHT printing.

A caption author is allowed to change the Print Direction attribute for a window to which text has already been written. It is recommended that upon receiving an SWA command with a new value for Print Direction compared with the direction of the last character sent to the window, all existing text in the window be cleared, and the cursor placed at the default cursor position (as if the window had just been defined).

Any text received prior to the new SWA command should not be redisplayed. Text received subsequent to the new SWA command should be written in the new print direction.

9.10.3 Scroll Direction

At a minimum, decoders shall support BOTTOM_TO_TOP scrolling.

For windows sharing the same horizontal scan lines on the display, scrolling may be disabled.

A caption author is allowed to change the Scroll Direction attribute for a window to which text has already been written. It is recommended that upon receiving an SWA command with a new value for Scroll Direction, all existing text in the window be cleared, and the cursor placed at the default cursor position (as if the window had just been defined). Any text received prior to the new SWA command should be removed from memory and not be redisplayed. Text received subsequent to the new SWA command should be written using the new scroll direction.

Caption authors should not rely on this operation, but should issue a ClearWindows command to empty the window being redefined.

9.10.4 Scroll Rate
At a minimum, decoders must support the same recommended practices for scroll rate as is provided for NTSC closed-captioning.

9.10.5 Smooth Scrolling
At a minimum, decoders must support the same recommended practices for smooth scrolling as is provided for NTSC closed-captioning.

9.10.6 Display Effects
At a minimum, decoders must implement the "snap" window display effect. If the window "fade" and "wipe" effects are not implemented, then the decoder will "snap" all windows when they are to be displayed, and the "effect speed" parameter is ignored.

9.11 DTVCC Section 8.4.11 - Window Colors and Borders
At a minimum, decoders need only to implement borderless windows with solid, black backgrounds (i.e., border type = NONE, fill color = (0,0,0), fill opacity = SOLID), and borderless transparent windows (i.e., border type = NONE, fill opacity = TRANSPARENT).

9.12 DTVCC Section 8.4.12 - Predefined Window and Pen Styles
Predefined Window Style and Pen Style ID's may be provided in the DefineWindow command. At a minimum, decoders should implement Predefined Window Attribute Style 1 and Predefined Pen Attribute Style 1, as shown in Table 23 and Table 24.

9.13 DTVCC Section 8.5.1 - Pen Size
All decoders shall support LARGE, STANDARD and SMALL pen sizes.

9.14 DTVCC Section 8.5.3 - Font Styles
Although a caption service provider may specify any one of 8 font styles using the **SetPenAttributes** command, decoders need only to implement a single font for caption text display.

- Decoders that implement more than one font but do not support a font style specified in the **SetPenAttributes** command should instead display the caption text in the most similar font available. In decoders with only one font (i.e., font style 0, the default), all caption text, regardless of the specified font style, will be displayed in the default font.

In decoders with more than one but less than eight fonts, unsupported font styles should be displayed using an alternate font, giving precedence to the spacing attribute of the indicated font style, if possible. For example, if the specified but unsupported font style is "monospaced with serifs", the best substitute would be another monospaced font, and the second-best alternative would be a proportionally spaced font with serifs. If the Cursive font style is not supported, an acceptable substitution is an italicized version of an available font.

All supported font styles may be implemented in any typeface which the decoder manufacturer deems to be a readable rendition of the font style, and need not be in the exact typefaces given as examples in Section 8.5.3.

9.15 DTVCC Section 8.5.4 - Character Offsetting
Decoders need not implement the character offsetting (i.e., subscript and superscript) pen attributes.

9.16 DTVCC Section 8.5.5 - Pen Styles
At a minimum, decoders shall implement normal, italic, and underline pen styles.

9.17 DTVCC Section 8.5.6 - Foreground Color and Opacity
At a minimum, decoders shall implement solid and flashing character foreground type attributes.

At a minimum, decoders shall implement the following character foreground colors: white, black, red, green, blue, yellow, magenta and cyan.

9.18 DTVCC Section 8.5.7 - Background Color and Opacity

Decoders need only implement solid black character backgrounds. It is recommended that this background is extended beyond the character foreground to a degree that the foreground is separated from the underlying video by a sufficient number of background pixels to insure the foreground is separated from the background.

9.19 DTVCC Section 8.5.8 - Character Edges

Decoders need not implement separate character edge color, opacity, and type attribute control. In this case, there is no separately controlled edge surrounding the body of characters.

9.20 DTVCC Section 8.8 - Color Representation

At a minimum, decoders shall support the 8 colors described in Table 27.

Color	Red	Green	Blue
Black	0	0	0
White	2	2	2
Red	2	0	0
Green	0	2	0
Blue	0	0	2
Yellow	2	2	0
Magenta	2	0	2
Cyan	0	2	2

Table 27 Minimum Color List Table

When a decoder supporting this Minimum Color List receives an RGB value not in the list, it shall map the received value to one of the values in the list via the following algorithm:

a) All one (1) values are to be changed to 0
b) All two (2) values are to remain unchanged
c) All three (3) values are to be changed to 2

For example, the RGB value (1,2,3) shall be mapped to (0,2,2), (3,3,3) will be mapped to (2,2,2) and (1,1,1) shall be mapped to (0,0,0).

Table 28 is an alternative minimum color list table supporting 22 colors.

Color	Red	Green	Blue
Black	0	0	0
Gray	1	1	1
White	2	2	2
Bright White	3	3	3
Dark Red	1	0	0
Red	2	0	0
Bright Red	3	0	0
Dark Green	0	1	0
Green	0	2	0
Bright Green	0	3	0
Dark Blue	0	0	1
Blue	0	0	2
Bright Blue	0	0	3
Dark Yellow	1	1	0
Yellow	2	2	0
Bright Yellow	3	3	0
Dark Magenta	1	0	1
Magenta	2	0	2
Bright Magenta	3	0	3
Dark Cyan	0	1	1
Cyan	0	2	2
Bright Cyan	0	3	3

Table 28 Alternative Minimum Color List Table

When a decoder supporting the Alternative Minimum Color List in Table 28 receives an RGB value not in the list (i.e., an RGB value whose non-zero elements are not the same value), it shall map the received value to one of the values in the list via the following algorithm:

a) For RGB values with all elements non-zero and different - e.g., (1,2,3), (3,2,1), and (2,1,3), the 1 value shall be changed to 0, the 2 value will remain unchanged, and the 3 value will be changed to 2.
b) For RGB values with all elements non-zero and with two common elements - e.g. (3,1,3), (2,1,2), and (2,2,3), if the common elements are 3 and the uncommon one is 1, then the 1 elements is changed to 0; e.g. (3,1,3) -> (3,0,3). If the common elements are 1 and the uncommon element is 3, then the 1 elements are changed to 0, and the 3 element is changed to 2; e.g. (1,3,1) -> (0,2,0). In all other cases, the uncommon element is changed to the common value; e.g., (2,2,3) -> (2,2,2), (1,2,1) -> (1,1,1), and (3,2,3) -> (3,3,3).

All decoders not supporting either one of the two color lists described above, shall support the full 64 possible RGB color value combinations.

9.21 Character Rendition Considerations

In NTSC Closed Captioning, decoders were required to insert leading and trailing spaces on each caption row. There were two reasons for this requirement:

a) to provide a buffer so that the first and last characters of a caption row do not fall outside the safe title area, and
b) to provide a black border on each side of a character so that the "white" leading pixels of the first character on a row and the trailing "white" pixels of the last character on a row do not bleed into the underlying video.

Since caption windows are required to reside in the safe title area of the DTV screen, reason a) is not applicable to DTVCC captions.

The attributes available in the **SetPenAttributes** command for character rendition (e.g. character background and edge attributes) provide flexibility to the caption provider when describing caption text in an ideal decoder implementation. However, manufacturers need only implement a minimum of pen attributes and font styles. Thus it is recommended that no matter what the level of implementation, decoder manufacturers should take into account the readability of all caption text against a variety of all video backgrounds, and should implement some automatic character delineation when the individual control of character foreground, background and edge is not supported, and when only a minimum number of font styles are implemented.

Decoder manufacturers should implement non-transparent backgrounds in such a way that the character's legibility is maintained even when the character occurs at the beginning or end of a line. This can be done by extending the background box (described in Section 8.5.7) beyond the character cell, by inserting a background-filled character space before the first character in a line and after the final character in a line, or by any other means that provides good legibility. This is consistent with recommendations in CEA-708-B Sections 9.18. The use of background-filled, as opposed to "fill color" filled improves legibility when "fill opacity" is not SOLID.

Decoder manufacturers should implement TRANSPARENT backgrounds and borders as transparent. This may include the character edges if the "border type" is set to NONE. Character edges may include transition areas surrounding the character pixels, which may be implemented to reduce ringing or other edge effects.

Decoder manufacturers should implement the transparent space as a completely transparent cell, except for transition areas at the beginning and end of a block of transparent cells, which may be implemented to reduce ringing or other edge effects.

9.22 DTVCC Section 8.9 - Service Synchronization

Service Input Buffers shall be at least 128 bytes in size. When a Delay command is encountered, the decoder should wait for the specified delay time to expire before processing any other service data. During the delay interval, incoming data for the active service should be buffered in the Service Input Buffer. When the delay interval expires, interpretation of the incoming data should be resumed.

9.23 DTV to NTSC Transcoders

Decoders which support NTSC outputs shall remove any CEA-608-C datastream bytes found in the DTVCC and place them on the NTSC output.

It is anticipated that receiver (decoder) manufacturers will develop devices (e.g., set-top boxes) which process an DTV stream and transcode it for display on NTSC monitors. The DTVCC command set is not necessarily transcodable to NTSC captions; i.e., there are DTVCC captions which have no NTSC equivalent.

Although receiver manufacturers are free to attempt an automatic transcode of the captions, there is no guarantee that the captions will appear as the caption provider intended. Caption providers apply many techniques to make the captions easy to read and as unobtrusive as possible over the underlying video. To maintain caption quality during an automated transcode process, a set of conversion rules would have to be defined which cover all possible window, pen and text attribute combinations.

Therefore, a separate NTSC caption channel was added to the picture **user_data()** (see Section 4.3). This channel allows caption providers to encode dual caption streams within the same programming. NTSC captions are under the complete control of the caption provider; and thus, no automated transcoding of captions is necessary.

9.24 Receivers Without Displays and Set-top Box (STB) Options

The NTSC outputs of DTV decoder devices (e.g. set-top boxes (STBs)) should contain the entire CEA-608-C datastream carried within a received DTV program, properly re-encoded into the NTSC video in accordance with CEA-608-C. DTV decoder devices may, in addition, process captions internally and provide an option for the user to select open captioned video on the NTSC output. When the program is transcoded to NTSC video after reception, the entire CEA-608-C datastream should be output in the NTSC video, thus ensuring proper operation of NTSC devices.

9.25 Use of CEA-608-C Datastream by DTV Receivers

DTV receivers may use the CEA-608-C datastream when CEA-708-C data is not available for providing closed captioning. If the CEA-708-C caption data is present, DTV receivers shall use it as the default closed caption format. DTV receivers should pass all XDS data through to any included NTSC outputs. Treatment of XDS data by DTV receivers may be bound by private agreements or government directives. Implementers are cautioned to read and understand all applicable agreements and directives. It is anticipated that XDS transmission is unlikely to continue long beyond the DTV transition.

10 DTVCC Authoring and Encoding for Transmission

This section briefly describes a DTV captioning "food chain". This is the path the captioning information takes from initial authoring intentions to being multiplexed into the ATSC emission bitstream. The content of this section is an example for information only. See SMPTE EG 43 for guidelines for station operation and caption handling.

10.1 Caption Authoring and Encoding

High quality captioning starts with the creation of the captioning intentions. This is a high level, generally editable, representation of how and when the captions should appear when rendered on the consumer receiver. SMPTE 12M time code is generally used for synchronization with picture. The output of the initial authoring process is generally a computer file that contains a list of time codes and the intention as to what the receiver should render when the picture, with the corresponding time code, appears on the display device. This computer file is typically editable. The file may be stored on a hard disc or floppy disc, and distributed by either computer networking techniques, or via floppy net. This process is illustrated in Figure 24.

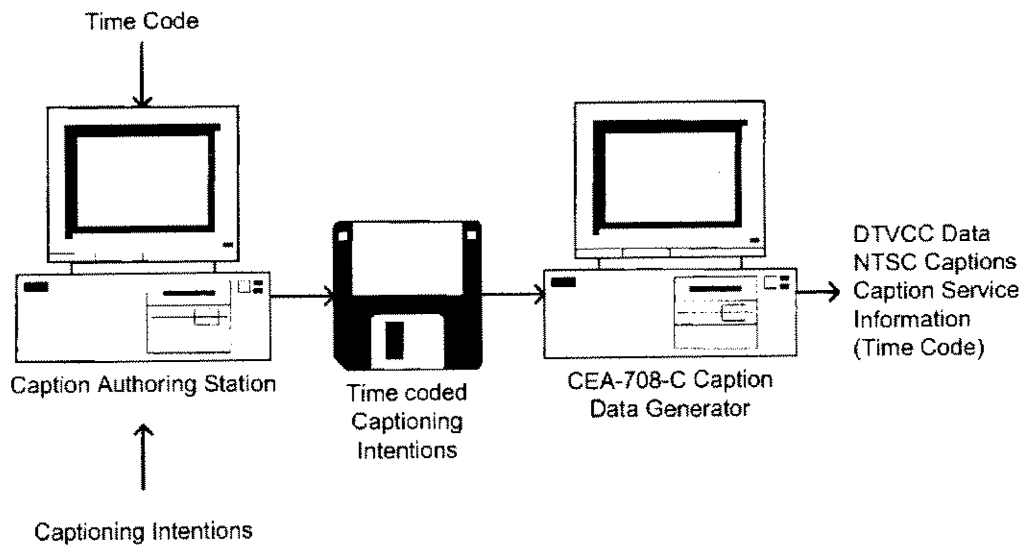
Time Code
Caption Authoring Station
Captioning Intentions
Time coded Captioning Intentions
CEA-708-C Caption Data Generator
DTVCC Data
NTSC Captions
Caption Service Information
(Time Code)

Figure 24 Caption Authoring and Encoding into Caption Channel Packets

The caption intentions are rendered into variable length *DTV Caption Channel Packets* (see Section 5). This rendering process is performed by an CEA-708-C caption data generator. This generator should also create a CEA-608-C datastream (as both forms are required to be emitted). This rendering process should consider the latency of the caption decoding process in the receiver, and thus should generate the packets pre-timed for transmission slightly early. Besides rendering the intentions into DTV Caption

Channel Packets, the generator should create the caption service information that will be used to create the caption service descriptors that are carried in both the MPEG-2 PMT and in the ATSC Program and System Information Protocol (PSIP) EIT.

When delivered to the consumer receiver by the ATSC DTV system, the NTSC and DTV caption data is carried in the MPEG-2 picture **user_data()**, where a fixed number of bytes is allocated for each frame. The allocation provides 9,600 bits per second of data capacity. The number of bytes carried with each picture frame is dependent on the picture frame rate (see table 3 in section 4.4.2). In the case of 29.97 or 30 fps pictures, there are 4 bytes of NTSC caption data and 36 bytes of DTV caption data carried per frame. The data rate of DTV Caption Channel Packets is equal to or lower than 9600 bps and so some zero padding is generally required. After the variable length DTV Caption Channel Packets are formed, it is necessary to parse these packets into the fixed length blocks that will be included with individual picture frames in the final emission multiplex. Some of these blocks will include zero padding so that the occupied bit rate is padded out to the full 9600 bps rate. This process is illustrated in Figure 25.

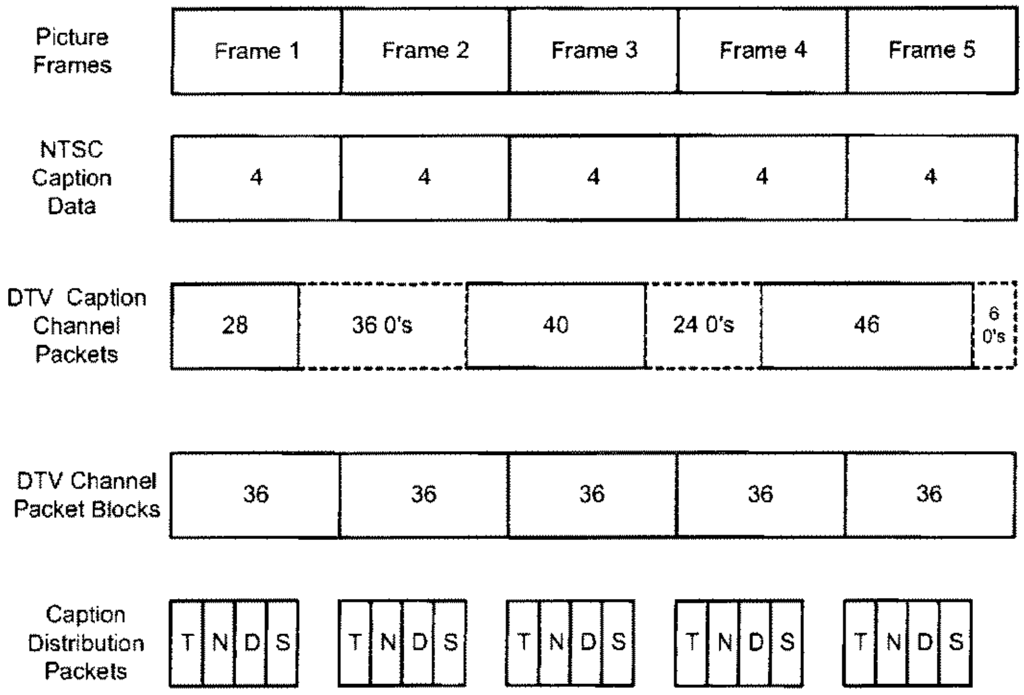

Figure 25 Relationship Between Caption Data and Picture Frames

Figure 25 illustrates five picture frame periods, at a 29.97/30 Hz frame rate. At this frame rate there are 4 bytes of NTSC caption data per frame. Three DTV Caption Channel Packets have been generated during the time of these 5 frames. These variable length packets have lengths of 28, 40, and 46 bytes. The DTV caption data is then formed into fixed length DTV Channel Packet Blocks that will be included in the corresponding MPEG-2 picture **user_data()** (see Section 4) area. In this 29.97/30 Hz frame rate example, these blocks have a uniform length of 36 bytes. The first of these blocks contain the 28 bytes of the first DTV Caption Channel Packet plus 8 bytes of zero padding. The second block contains 28 bytes of zero padding followed by the first 8 bytes of the second DTV Caption Channel Packet, and so on. The result of this process is 36 bytes of DTV captioning data per picture frame. The final line in this figure shows the formation of all of the captioning data into *Caption Distribution Packets* that will be described in Section 11. These packets have a 1:1 relationship to video frames, and can include time code (T), the NTSC caption data (N), the blocked and zero padded DTV caption data (D), and the caption service information (S).

10.2 Monitoring Captions

Caption data may be monitored at various points along the distribution chain. To monitor the captions as they will be displayed on a consumer receiver, and to evaluate video/caption synchronization, it is necessary to decode the caption data after it has been rendered into a form where there is an association of sets of caption data bytes with particular video frames.

10.3 Encoder Interfacing

DTVCC and NTSC captions are multiplexed into the picture **user_data()** in the MPEG-2 video elementary stream. This multiplexing is generally done within the MPEG-2 video encoder and so the caption data must be delivered to this encoder. A Caption Service Descriptor is included in the PMT and in the EIT and so caption service information is delivered to the PSIP generator, and to the MPEG-2 PMT generator. (It is possible that both PMT and EIT would be created by a common functional block.) It is likely that all of the caption information will arrive at the encoding system on a single interface connection, and be distributed to the various destinations by looping the interface connection to multiple functional blocks. (Some systems may extract the data for the CSD from video data. One method for subsequent communication of the extracted data is ATSC A/76.) See Figure 26.

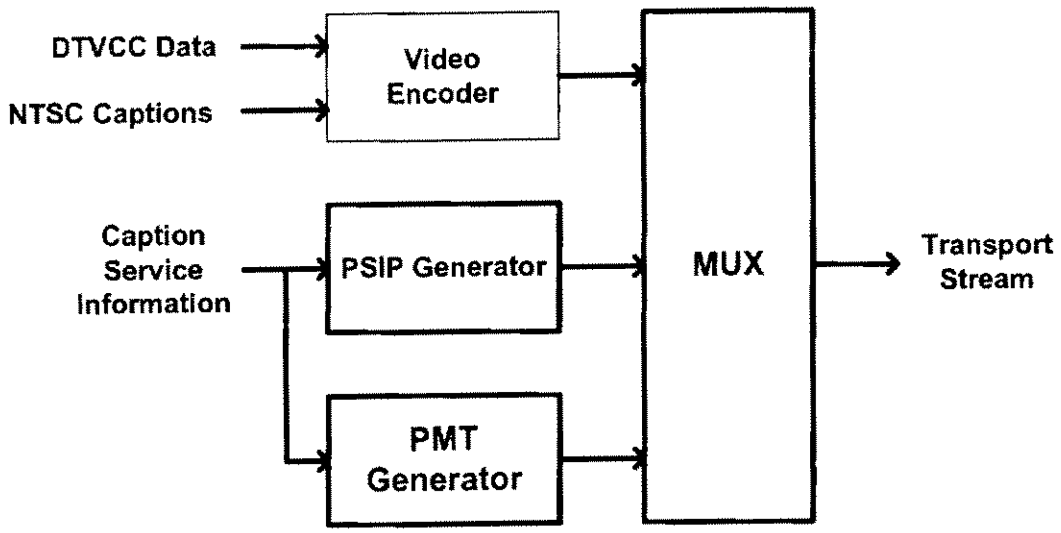

Figure 26 Interface of Caption Data to ATSC Emission Encoding Equipment

Several types of interfaces for the caption data may exist. These include a serial data format carried on an asynchronous RS232 like interface, embedding into an AES3 (digital audio interface) data stream, or embedding into SMPTE 259M or SMPTE 292M serial digital video streams. For purposes of interoperability it is useful if there is as much commonality as possible to the data format on these different types of interfaces.

Annex A Possible Decoder Implementations (Informative)

Figure 27 illustrates one example of a decoder implementation for processing the PMT, EIT and User Data in the DTVCC Transport Stream. It shows the separate paths of the Service Descriptors in the PMT and EIT and the service data in the picture **user_data()** bits.

NOTE—Different block diagrams may exist for other implementations.

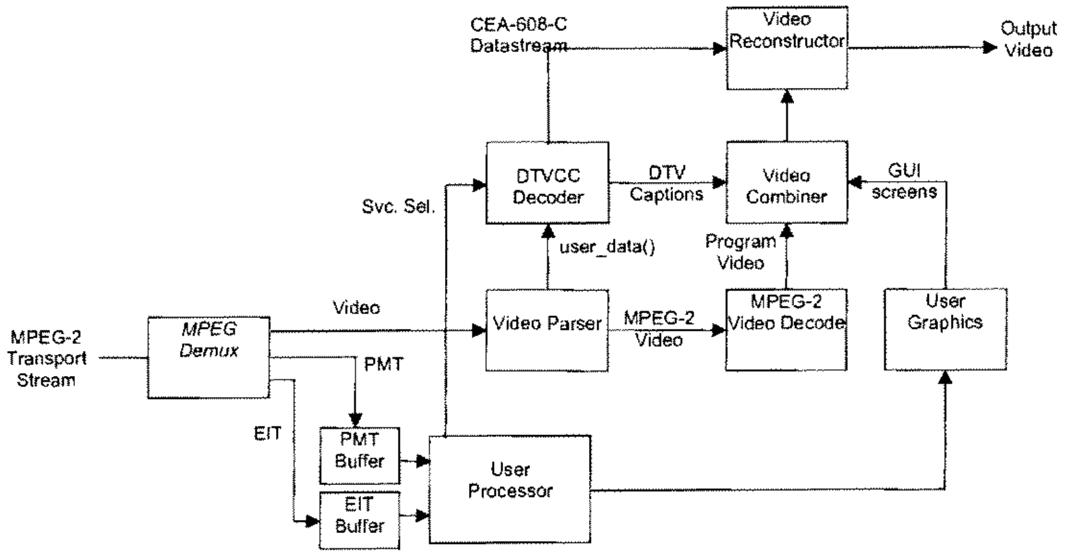

Figure 27 DTVCC Transport Stream Decoder

Annex B Transmission

B.1 Interpretation of Transmission Syntax

CEA-708-B Section 4.4.1 provides guidance for the use of **cc_valid** and **cc_type**. These values should allow creation of any NTSC caption signal. The following discussion provides examples of the use of **cc_valid** and **cc_type** and how they can be applied to a variety of captioning situations.

B.1.1 Representation of NTSC Caption Signals

The DTVCC Transport Channel is designed to convey a CEA-608-C datastream through a DTV Bitstream. The CEA-608-C datastream can be conveyed with total fidelity if the proper methods are used, as shown in the following examples.

Possible outputs on line 21 are:

a) No waveform on line 21
b) Waveform with zero-filled data
c) Waveform with invalid caption data (bad parity, etc.)
d) Waveform with valid (including "null" data)

Each of these outputs can be created by the proper choice of **cc_valid**, **cc_data1** and **cc_data2**. For this discussion to apply to field 1 data, **cc_type** equals 00, for field 2 data, **cc_type** is 01.

If **cc_valid**==0, no waveform may be generated for the specified field of line 21. However, if field 2 is generated, field 1 shall be generated. CEA-708-B Section 4.4.1 satates: "If **cc_valid** = 0 then the run-in clock and start bits are not required to be present, except if they are present on field 2 they are required to be present on field 1. This is a method of generating output "a".

Waveform group "b", a waveform with zero-filled data, can be created by making **cc_valid** be one, and by making the **cc_data()** bytes be 0x00. Output "c" is similar, using other even-parity data. CEA-608-C Section 8.3 states that the data shall be encoded with "ODD parity...." These two types of waveforms contain invalid data, and appear to violate this requirement. They should not be generated for normal caption encoding. Clearly, if an ATSC encoder receives invalid caption data, it is permitted to copy that same data into the MPEG-2 stream. Further, if a decoder receives invalid data in the ATSC **user_data()** construct, it can encode that (invalid) data onto NTSC line 21. The transmission of data does not require modification of that data, even if it does not have valid parity.

The final waveform, "d", is produced by setting **cc_valid** to 1, and the **cc_data()** bytes to odd-parity values (when **cc_type** is set to 00 or 01).

For example, when the input is two CEA-608-C "nulls", the **user_data()** should contain **cc_valid** = 1 and **cc_data_1**, and **cc_data_2** both equal to 0x80.

Input signals and encoders that create valid CEA-608-C datastreams abide by the CEA-608-C Section 8.2 and Annex D.7 requirement that field 1 contain valid data (including nulls) when field 2 data is present. Therefore, when encoding them into the DTVCC Transport Channel, **cc_valid** will be 1 for **cc_type** equal to 00 (field 1) if **cc_valid** is 1 for **cc_type** equal to 01 (field 2).
Despite this description of requirements for input signals, decoders cannot make this assumption, but must process signals as they are received.

CEA-708-B Section 4.4.3 describes the encoding of a CEA-608-C datastream, and how it relates to the **picture_structure** defined in the MPEG-2 **picture_coding_extension**. The example in Annex C shows how the **repeat_first_field** values affect the format of the CEA-608-C datastream.

Annex C DTVCC Channel Packet Transmission Examples

This is an example of three successive pictures with alternating values of **repeat_first_field**. The frame_rate_value is 29.97 or 30, and **progressive_sequence** is 0. These pictures are shown after being rearranged into display order. (See Annex D).

C.1 PICTURE 1: picture_structure = 11, top_field_first = 1, repeat_first_field = 1

The DTV caption data clears and hides all windows for service #1, and hides all windows for service #2. This shows how service blocks can share a DTVCC Packet.

This picture illustrates two paragraphs of 4.4.2. "... for pictures with **top_field_first** == 1, **cc_type** 01 shall precede **cc_type** 00." and "... **cc_type** 00 and **cc_type** 01... should alternate..."

There are 30 data pairs in this **user_data()**, as required for this picture's header values.

	cc_valid	cc_type	cc_data1	cc_data2
1	1	01	CEA-608-C datastream field 2	CEA-608-C datastream field 2
2	1	00	CEA-608-C datastream field 1	CEA-608-C datastream field 1
3	1	01	CEA-608-C datastream field 2	CEA-608-C datastream field 2
4	1	11	DTVCC Pkt Header = 0x05	DTVCC Pkt Data = 0x27
5	1	10	DTVCC Pkt Data = 0x8A	DTVCC Pkt Data = 0xFF
6	1	10	DTVCC Pkt Data = 0x88	DTVCC Pkt Data = 0xFF
7	1	10	DTVCC Pkt Data = 0x42	DTVCC Pkt Data = 0x88
8	1	10	DTVCC Pkt Data = 0xFF	DTVCC Pkt Data = 0x00
9	0	00	0x00 = padding	0x00 = padding
...				
30	0	00	0x00 = padding	0x00 = padding

Table 29 DTVCC Channel Packet Transmission Example A

C.2 PICTURE 2: picture_structure = 11, top_field_first = 0, repeat_first_field = 0

The DTV caption data had no information to send, but at least one data pair with **cc_type** = 10 or 11 is required to end the CEA-608-C datastream. In this case, the syntax used is defined in Section 4.4.1, and indicates the end of a DTV Caption Channel Packet. The data values (0xXX) mean "don't care".

	cc_valid	cc_type	cc_data1	cc_data2
1	1	00	CEA-608-C datastream field 1	CEA-608-C datastream field 1
2	1	01	CEA-608-C datastream field 2	CEA-608-C datastream field 2
3	0	11	0xXX = END OF PACKET	0xXX = END OF PACKET
4	0	00	0x00 = padding	0x00 = padding
5	0	00	0x00 = padding	0x00 = padding
6	0	00	0x00 = padding	0x00 = padding
7	0	00	0x00 = padding	0x00 = padding
8	0	00	0x00 = padding	0x00 = padding
9	0	00	0x00 = padding	0x00 = padding
...				
20	0	00	0x00 = padding	0x00 = padding

Table 30 DTVCC Channel Packet Transmission Example B

C.3 PICTURE 3: picture_structure = 11, top_field_first = 0, repeat_first_field = 1

In this picture, the DTVCC information contains two DTVCC Caption Channel Packets.

The first contains information for service #1; it selects (hidden) window #0, writes "This is a test" into it, and then displays it.

The second does some operation to service #2. The data for service #2 is not shown, but includes at least two service blocks (only start of first service block is shown).

Encoders may place each service in a unique DTVCC Caption Channel Packet, if desired.

	cc_valid	cc_type	cc_data1	cc_data2
1	1	00	CEA-608-C datastream field 1	CEA-608-C datastream field 1
2	1	01	CEA-608-C datastream field 2	CEA-608-C datastream field 2
3	1	00	CEA-608-C datastream field 1	CEA-608-C datastream field 1
4	1	11	DTVCC Pkt Header = 0x4A	DTVCC Pkt Data = 0x31
5	1	10	DTVCC Pkt Data = 0x80	DTVCC Pkt Data = 0x54
6	1	10	DTVCC Pkt Data = 0x68	DTVCC Pkt Data = 0x69
7	1	10	DTVCC Pkt Data = 0x73	DTVCC Pkt Data = 0x20
8	1	10	DTVCC Pkt Data = 0x69	DTVCC Pkt Data = 0x73
9	1	10	DTVCC Pkt Data = 0x20	DTVCC Pkt Data = 0x61
10	1	10	DTVCC Pkt Data = 0x20	DTVCC Pkt Data = 0x74
11	1	10	DTVCC Pkt Data = 0x65	DTVCC Pkt Data = 0x73
12	1	10	DTVCC Pkt Data = 0x74	DTVCC Pkt Data = 0x89
13	1	10	DTVCC Pkt Data = 0x01	DTVCC Pkt Data = 0x00
14	1	11	DTVCC Pkt Header = 0x91	DTVCC Pkt Data = 0x5A
...				
30	1	10	DTVCC Pkt Data = 0xXX	DTVCC Pkt Data = 0xXX

Table 31 DTVCC Caption Channel Transmission Example C

Annex D Transmission Order and Display Process Examples

The transmission order of pictures in an MPEG-2 stream does not need to be identical to the display order. Use of "B" pictures allows reordering of the image sequence before transmission, with the complementary reordering after the bitstream is converted back to images by the MPEG-2 decoder.

Decoders normally process captions as part of the display process, not the MPEG-2 decoding process. The "NOTE" in Section 4.4.1 explains this, stating that captions are interpreted in display order. This is true of both DTVCC Caption Channel Packets and the CEA-608-C datastream. The display process is a broad term. For example, it could be realized by sending pixel data to a Liquid Crystal Display (LCD) screen, sending RGB waveforms to a video display or sending an NTSC formatted video signal to a Radio Frequency (RF) modulator. As part of these example display processes, captions could be rendered onto the same LCD array, keyed into the RGB waveforms, or modulated into an CEA-608-C waveform and inserted in line 21 of the NTSC signal.

CEA Document Improvement Proposal

If in the review or use of this document, a potential change is made evident for safety, health or technical reasons, please fill in the appropriate information below and email, mail or fax to:

Consumer Electronics Association
Technology & Standards Department
2500 Wilson Blvd.
Arlington, VA 22201
FAX: 703 907-7693
standards@ce.org

Document No.	Document Title:
Submitter's Name: Submitter's Company:	Telephone No.: FAX No.: e-mail:
Address:	
Urgency of Change: Immediate: ☐	At next revision: ☐
Problem Area: a. Clause Number and/or Drawing: b. Recommended Changes: c. Reason/Rationale for Recommendation:	
Additional Remarks:	
Signature:	Date:
FOR CEA USE ONLY Responsible Committee: Chairman: Date comments forwarded to Committee Chairman:	

CEA
Consumer Electronics Association

cc:Mail for: Richard Lowell

Subject: URL Codes Working Group
From: "Broberg; David-MSM" <DBroberg@bigscreen.mea.com> at ccmail 1/30/97 2:56 PM
To: Richard Lowell at SEL-SD-TVA
To: "'Schumann; Tom -Philips'" <schumant@knox.pcec.philips.com> at ccmail
To: "'Kempter; Paul -Nielsen'" <paulk2@aol.com> at ccmail
cc: "'Hanover; George - CEMA'" <ghanover@eia.org> at ccmail

Greetings all,

I have enclosed the updated draft of the proposal for URL codes in line 21 XDS.
I have made the changes from my notes at the last meeting in Las Vegas.

I don't have any idea on recommendations for refresh rates. Do any of you?

Please review the enclosed document and suggest any changes as necessary.
Thanks.
-dB

```
>*****************************************************
>David Broberg,
>Manager, Internet Business Development
>Mitsubishi Consumer Electronics America
>6100 Atlantic Boulevard/   Norcross, GA 30071
>Phone:  (770) 734-5449/   Fax:  (770) 734-5352
>Email:  dbroberg@bigscreen.mea.com
>URL:   www.mitsubishi.com
*****************************************************
```

BTW
I don't have Paul Kempter's office e-mail, if you do, could you please forward this to his office.

PROPOSAL TO CEMA R-4.3
for New XDS Packets for EIA-608

October 11, 1996

By David Broberg, Mitsubishi Consumer Electronics America, Inc.

Executive Summary:
Proposed below are six (6) new XDS packets that may be used within the EIA-608 system for Extended Data Services. These packets define a standard method to include program related Internet Universal Resource Locators (URL's) within the line-21 data system. These URL's may be used by receiving devices to permit the linking of television programs with related content on the Internet so that these content from these Internet services may be combined, mixed or shared by the receiving device. Mitsubishi Consumer Electronics America, Inc. (MCEA) claims intellectual property rights in the implementation and execution of these functions within the receiving device. In accordance with EIA/CEMA by laws, MCEA will offer license to such technology to all interested parties in non-discriminatory terms.

New Packet Overview:
Six new XDS packets are described, each may be used within the Current or Future class as defined in the section 6.4.1 and 6.4.2 of EIA-608. Each packet consists of a control code pair followed by a payload of 2 or more Informational Characters followed by the End/Checksum pair. Each packet may be interleaved, interrupted and continued based on existing XDS recommended practices. In accordance with section 6.2.2, each packet may have more than 32 informational characters, but will always include nulls characters to fill the packet to an even number of informational characters, and will always end with the End/Checksum pair.

------------------------------- *to be inserted in section 6.5.1 of EIA-608* -------------------------------

18h Program's Web Site

This packet contains a variable even number of Informational characters that define the Universal Resource Locator (URL) of the program's designated World Wide Web address. Each character is within the EIA-608 ~~an ASCII~~ character set and in the range of 20h to 7Fh. The variable size of this packet allows for efficient transmission of URL's of any length. A change in the received Current Class Program Name is interpreted by the XDS receivers as the end of the valid period of this link address. The format of this packet is shown in the following example:

Start	PID	Payload	End	Checksum
01h	18h	http://www.programhomepage.com	0Fh	XXh

19h Sponsor's Web Site

This packet contains a variable even number of Informational characters that define the Universal Resource Locator (URL) of one of the program's sponsors' designated World Wide Web addresses. Multiple Sponsor Web Site packets may be associated with each program. The Sponsor's Web Site packet may be transmitted during any portion of the program including, but not limited to the sponsor's break. Each character is within the EIA 608~~an ASCII~~ character set and in the range of 20h to 7Fh. The variable size of this packet allows for efficient transmission of URL's of any length. A change in the received Current Class Program Name is interpreted by the XDS receivers as the end of the valid period of this link address. The format of this packet is shown in the following example:

Start	PID	Payload	End	Checksum
01h	19h	http://www.sponsorhomepage.com	0Fh	XXh

1Ah Sponsor's Name

This packet contains a variable even number, 2 to 32, of Informational characters for the name of program's sponsor associated with the most previously received sponsor link. Multiple sponsor names may be associated with each program. The Sponsor's Name packet may be transmitted during any portion of the program including, but not limited to the sponsor's break. Each character is within the EIA-608~~an ASCII~~ character set and in the range of 20h to 7Fh. The variable size of this packet allows for efficient transmission of names of any length. A change in the received Current Class Program Name is interpreted by the XDS receivers as the end of the valid period of this sponsor's name. The format of this packet is shown in the following example

Start	PID	Payload	End	Checksum
01h	1Ah	Sponsor Products Company	0Fh	XXh

1B$_h$ Related News Group Link

This packet contains a variable even number of Informational characters that define the Universal Resource Locator (URL) of the program's related Internet News Group address. Each character is within the EIA 608 ~~an ASCII~~ character set and in the range of 20h to 7Fh. The variable size of this packet allows for efficient transmission of URL's of any length. A change in the received Current Class Program Name is interpreted by the XDS receivers as the end of the valid period of this link address. The format of this packet is shown in the following example:

Start	PID	Payload	End	Checksum
01h	1Bh	alt.fanclub.group	0Fh	XXh

1C$_h$ Related Chat Link

This packet contains a variable even number of Informational characters that define the Universal Resource Locator (URL) of the program's designated chat site address. Each character is within the EIA-608~~an ASCII~~ character set and in the range of 20h to 7Fh. The variable size of this packet allows for efficient transmission of URL's of any length. A change in the received Current Class Program Name is interpreted by the XDS receivers as the end of the valid period of this link address. The format of this packet is shown in the following example:

	Start	PID	Payload	End	Checksum
	01h	1Ch	http://programchat.com	0Fh	XXh
or	01h	1Ch	irc-2.programchat.com	0Fh	XXh

1D$_h$ Mail-to Link

This packet contains a variable even number of Informational characters that define the designated Internet E-Mail address for the program. Each character is within the EIA-608 ~~an ASCII~~ character set and in the range of 20h to 7Fh. The variable size of this packet allows for efficient transmission of e-mail addresses of any length. A change in the received Current Class Program Name is interpreted by the XDS receivers as the end of the valid period of this link address. The format of this packet is shown in the following example:

Start	PID	Payload	End	Checksum
01h	1Dh	comments@program.network.com	0Fh	XXh

6.6.2 XDS Packet Usage Recommendations:

6.6.2.16 Program's Web Site

The payload of this packet will enable receiving devices with internet access and web-browsing capabilities to link to a web site as designated by the program provider. When an XDS receiver recognizes such a packet and it passes checksum and all parity tests, an indication may be given (or is available) on the receiver to inform the user that a program link is available. When the user of the receiving device chooses to activate or select the link, the receiving device will initiate the connection to the designated internet address. To make best use of the limited bandwidth available on the XDS channel, there is no HTML or web-page content transmitted within this packet, only the address. Receiving devices may also cache (or store) these links and index them by channel, specific program or time of day.

6.6.2.17 Sponsor's Web Site

Of course programs that include paid sponsors will most often include many unrelated sponsors for the same program. This protocol will allow several sponsors to each identify their own unique web-site address. The insertion of these sponsor links will require that each unique sponsor URL be followed by a sponsor name tag. When an XDS receiver recognizes a sponsor link packet and it passes checksum and all parity tests, an indication may be given (or is available) on the receiver to inform the user that a sponsor link is available. When the user of the receiving device chooses to activate or select the link, the receiving device will initiate the connection to the designated internet address. To make best use of the limited bandwidth available on the XDS channel, there is no HTML or web-page content transmitted within this packet, only the address.

6.6.2.18 Sponsor's Name

The sponsor's name packet is used to support multiple sponsors for the same program. As each sponsor's link packet is transmitted, it is followed by the sponsor's name packet (before any new sponsors link packets may be sent). Upon receiving these packets in the receiver, a list is constructed and stored that ties the designated name with the previously sent link. The user of the receiving device is then able to make a selection among several active (stored) sponsors, to link to the desired one anytime during the program. This ensures that each sponsor of a program has an equal opportunity to make use of the link in the receiver, without one sponsor bumping the previous one. Of course, receiving devices may also cache a series of sponsor links each associated with or indexed to a channel, a unique program or time of day.

6.6.2.19 Related News Group Link

| In the interest of~~in~~ conserving bandwidth of the XDS system, links to News Groups should first be placed within the program's designated web site. If no designated web site exists for the program, this packet may be used instead to allow a direct link to a designated or related news group associated with the program. It will be used to enable receiving devices with news readers to link directly with this internet site. Like the previous packets, this packet may also be cached or stored by the receiving device.

6.6.2.20 Related Chat Link

In the interest in conserving bandwidth of the XDS system, links to Chat sites should first be placed within the program's designated web site. If no designated web site exists for the program, this packet may be used instead to allow a direct link to a designated or related chat site associated with the program. It will be used to enable receiving devices with web browsers or internet relay chat software to link directly with this internet site. Like the previous packets, this packet may also be cached or stored by the receiving device. Multiple formats are permitted within the payload section to permit chat sites that are IRC protocol, HTML protocol or others.

6.6.2.21 Mail-to Link

This packet enables the transmission of an Internet E-mail address that is associated with the current program. Receiving devices that support Internet E-mail may recognize this packet, store it, or use it to inform the user that electronic replies are possible to the provider of this program. When captured by the receiver, the payload of this packet is inserted as the address for reply messages. It is also possible that the receiver may store this address in a permanent address book for later use.

EIA TDSS meeting Las Vegas 1/7/97 ①

10:00
000 introductions Jean Johnson – CEMA

030 minutes review

264 agenda.

292 ATVCC – critical path item The selling (Intro) of ATV sets could be held up
305 708 update because closed captioning standard may be lagging
working 708 and min req. doc on web site. Because of limited group involvement! little outside review
325 ballot for information as Informal ballot. to get comments back from full committee.
website http://village.ios.com/~markdale/atvcc
circulate to 4.3 and 4 for comment by reciever manufacturers & others.

410 77 pages = doc + RP + minimum requirement,

525 Responses back by Feb. 21

Next meeting March 24 princeton NJ
R-4 March 26
R 4.1, 4.6 25

537 1/4 quarter 98 optomistic – ATV product more likely 2nd to 3rd quarter
600 Jan 31 out by – sent out no later → March 7 – back by
~~Feb 21~~ ballots due ~~back date~~

March 14 – comments to Amnon WG.
14 – 24 put info together & present status of draft for full ballot.

640 Simulation status / testing
Critical will be the
620 ~~Also need to know~~ simulator and implemtation issues and can be done in parallel with informal ballot.

↳ Mid Feb, early March. Simulator should be completed

for Implementation into GA scheme will require additional funding.
688 current funding will be discontinued in March. by Department of Education. so work should be done by March.

1a

709 couple modules have been created

script program to create text

~~to~~ another ~~to~~ display on computer monitor.

Mark Dale is involved in coding and how informat. is sent to the display mechanism.

extra. is the ~~to~~ required details needed to work with an ATV station.

↓307 ATVCC WG meeting, lucent, baseband simulator, ATVCC WG. Phillips, Sarnoff, Berny Lechner at WRC #99 Jim McKinney to host, EEG, NAB, Ralph Justus, Gil Hanaras the video

most import. (To ensure proper transport and ensure 708 is working along with the stream. Construct the transport of 708 data properly encoded into the video stream and properly decoded from it. and display how ever possible as soon ~~as~~ possible

secondly. Simulation is a perceptive ~~#~~. Simulation on an actual overlay of an HD is of secondary importance can be taken ~~up~~ separately

↓ 277 What's important is for TV manu. to look at on what ever kind of display what the control codes do to the delievered closed captioning. so they can interpret and setup their display in their HD Televisions. that is most suitable to them.
Scrolling, placement, size, fonts, colors to get idea what is available for them to put in CC decoder's.

↓240 Still need to resolve.
How the insertion of captions will work. In terms of splice points. and the transmission in bitstream vs display order. in MPEG packed

2a

162. Design cycle.

TV manufacturers suggestion.

July 1 is new model Introduction.

for a new model FCC ruling would have to be 18 months before July 1

Zenith design cycles ~~and~~ are developed around product Introduction cycles. ~~which are in spring.~~ May of every year is the time period. when new products are introduced. and shortly after that these products go into production.

230 the FCC wants to corrdinate the rule making ~~to~~ the general design cycle.

The clock should start ticking when the technical standard is complete and approved not necessarily when the FCC makes a rule.

310 the president ~~asked for~~ suggested a trial period of 9-10 mnths

But the NPRM could be issued in 2-3 months

~~The FCC would wait~~

this could clear up some of the issues but surely the FCC would wait til the end of the trial period before the commission makes its final decisions.

The final system has to in law before receiver manufact have the certainty of knowing that they can go ahead with the finalization of the design.

561 Could require multiple systems
MPAA
TV Rating G.
B category 4 level,

↓223 The goal is to by next meeting March 24 to have the 1st transmission of 708 data by WHDTV ②

↓181 Research. CEMA
Manufacturer/Consumer - EIA may not be doing

↓175 Caption service ~~index~~ directory - where it goes

In PMT always. until change in caption stream ⇒
? In service zero of the closed caption data stream or in a separate descriptor carried in the MPEG PMT, - disagreement between GI and Thomson on which way to go. did reach a compromise.

↓141 9600 bit always be available
plus 480bps for 608 bitstream

↳ so the receiver doesn't have to constantly process the PMT. The ~~document~~ ATSC Document T3S8 175 will need to modified.

↓120 review - Standard release time line discussion

↓039 SDTV 608 decoder a minimum.
↓009 708 is to be ready for ballot at next meeting.
~~Even~~ with no negative ~~ballots~~ comments ~~then~~ an approve standard won't be available until July.
tape 2 non ANSI (30 day) interim standard.
017 1st (hopefully last) ballot returned and available for resolution by the meeting after next.

070 schedule review.
March 24 - will send out a 30 day interim standard and maybe resolve comments before the May 15 meeting.
103 repeated: informal ballot and on-line testing before March 24
after March 24 formal committee ballot 15 EIA by April 1
return responses by May 1.
resolve comments by May 15.

119 Lunch
125 V-chip Rick Engleman TV ratings
TV industry group did publically announce their rating system in December and did meet with the commission to discuss that plan. The TV industry group is to submit something in writing ~~ratings plan information with~~ to the FCC within a few weeks. FCC will put it out for public (notice) comment for a 60-90 days to see what the reaction is. Then after the initial public comment period. there would be a separate notice of proposed rulemaking covering the technical standard issues, respect to requirements on TV manufacturers.

575. Canada tests are again in progress
Now carried in XDS and still a content and not a age rating.
Blocking by scene was found to be a problem by viewers.
and it was found to block the entire show.

677 21 line part reads XDS and separates out the rating packets
(field two data slicer

709 Public Notice – maybe NPRM or other
↓783 The law is supportive of VBI line 21, it could be XDS or something else.
↓ 751 ATV futher notice. (Kelly) should be a sepate issue.
↓ 745 Content Advisory Issues
How to handle with the FCC.

↓573 This is the system the Television industry is going to use
Will use no other rating system. If the government tries to force something else they would take it to court.
↓ 1–4 is capabile of supporting the valenti codes
↓ 355 – XDS delivery
– two bytes of data including current MPAA data · EIA 608 A
– 3 sec. repeat. rate
– Field 2 captions still have priority.
↓ 070 Summary

tape 3
000 testing purpose / goal
107 ~~deter~~ Summary
determine what field testing is needed
use current class 05H packet,
leave MPAA rating as is.
use repeat rate of 3 sec. or longer
field two captions have priority over all other data
this system will be able to support a rating scheme such as not proposed by Valenti Group

268 patents

Exhibit 15 Page 425

SONY0073800

CONFIDENTIAL

270 Additions to 608 updates
1. Channel mapping Auto Cable Installation - 728 some areas still need to be resolved by another teleconference
2. URL packet proposal
3. CGMS-A

776 maybe important for cable ready spec. So should go out as IS Committee ballot after next meeting,

419 DVD caption output on line 21
philips player will pass line 21

510 URL packet Mitsubishi proposed

WG
Lowell
Schuman
Broberg

608 character set available only.
ASCII
Mitsubishi patents
HTML patents (hyper links.)

Determine if URL draft is ready ~~to be~~ submitted to R-4

725 Ballot resolution. 608 A
responded to yes conditional voter
19 votes, 2 no (~~HBO~~, Parent Technologies, Zenith)
1st 17
2nd 19
Resolution task force has sent out letters, but have not heard back from ~~anybody~~. Parent Technologies, Committee disagrees.

521 SMPTE (subtitles) WG. - There have been two meetings so far
Standard for subtitles on DVD

- or send control codes WRSAME.

478 ATSC Emergency messaging - Voluntary
Define the syntax for a data byte ~~for~~ in video user bits in ATV called EM data. It has a data rate of 300 bits per sec. It could carry 708 ~~text~~ style data for sending a message, GI says that its low data rate is not sufficient to satisfy emergency alert requirements, GI wanted to define a set of instructions that would point to another data channel contained in another PID.
Kelly will draft a ~~something~~ straw man proposal for the syntax for this proposal to get feedback from the working group and report to ATSC.
GI thinks that it might be mandatory - auto on, and interrupt program.

160 Mission Statement - Done

155 New Business

153 Next Meeting March 24 March 26 - R4 Hitachi accessible by taxi from train station Princeton NJ 10am
May middle

Exhibit 15 Page 426

SONY0073801

CONFIDENTIAL

5a.

"...If someone wants to deliver the data to a receiver and the receiver ~~[illegible]~~ wants to display it

it nonsynchronous data, slam some serial data into the input of the back of an encoder

you have to tell me what you want to do with it once you get it, and that depends on the semantics.

EIA Registry of packets with XDS
Undefined packets indented for use in closed systems

ATV Test station in Washington next week - ATVCC WG.
or the ~~following~~ week after that. meeting

EBS to EAS transition

for broadcasters it is a one year transition period. which started January 1st of this year.
you have to have the equipment this year
and by January 1 1998 you must take the old equipment out of service.

Cable starts July 1 of this year.

EAS carries FSK ~~tones.~~ It carriers some data with it,
(FCC name ASK
that tells things like who originated ~~it~~ the message, how long it is, what type of alert it is, what kind of counties that kind of thing.

Federal plans
State plans
County
City

Federal over rides others. Federal alert means the big one is coming.

Broadcaster have to have the equipment in place and start run the tests and alerts ~~as~~ of January 1 of this year

In late December ABC news did a story on the demise of of the EBS system. They used the EBS tone and set off receivers all over the place.

The old EBS equipment stays in place for a period of one year. and is taken out January 1 98. The tones sunset the year 2000. The requirement is to get the president's voice on the air.
Also is a PEP system that is AM based.

FCC requirement for Emergency notification requires both aural and visual. program is interuppted like inserting an add, Define the packet, syntax and the sematics for the packet